

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN 1



BÁO CÁO
MÔN HỌC: KIẾN TRÚC VÀ THIẾT KẾ PHẦN MỀM
NHÓM LỚP HỌC: 05
Giảng viên: Trần Đình Quê

Sinh viên: Lê Hồng Ánh
Mã sinh viên: B20DCCN083
Lớp: CNPM05
Nhóm: 01

Mục Lục

| | |
|--|----|
| Câu 1: Mô tả bằng dạng Bảng và ngôn ngữ tự nhiên các chức năng tương ứng với các actor và phi chức năng của Hệ thống ecomSys | 5 |
| 1. Mô tả ngôn ngữ tự nhiên | 5 |
| Customer Module | 5 |
| Manager Module | 5 |
| Book Module | 5 |
| Mobile Module..... | 5 |
| Clothes Module | 5 |
| Search Module | 5 |
| Cart Module | 6 |
| Order Module..... | 6 |
| Payment Module | 6 |
| Shipment Module | 6 |
| 2. Mô tả bằng dạng Bảng..... | 6 |
| Câu 2. Vẽ biểu đồ use case tổng quát và biểu đồ use case chi tiết cho từng chức năng dịch vụ | 8 |
| Usecase tổng quát: | 8 |
| Usecase chi tiết: | 9 |
| Usecase Login: | 9 |
| Usecase Search..... | 9 |
| Usecase Manage Cart | 10 |
| Usecase Order and Payment | 10 |
| Usecase Customer history ordered..... | 10 |
| Usecase Manage profile | 11 |
| Usecase Manage Customer..... | 11 |
| Usecase Report..... | 12 |
| Usecase Manage Product..... | 12 |
| Usecase Manage Order..... | 13 |
| Usecase Manage shipment..... | 13 |
| Câu 3 Vẽ biểu đồ phân rã Hệ ecomSys thành các service và các tương tác giữa các dịch vụ (sử dụng quan hệ << use>> trong UML) | 14 |
| Câu 4 Trình bày các dạng communication giữa các service với nhau (synchronous và asynchronous) với code và ví dụ | 15 |
| Giao tiếp Đồng bộ (Synchronous Communication):..... | 15 |
| 1. HTTP/HTTPS Request-Response: | 15 |

| | |
|--|----|
| Giao Tiếp Không Đồng Bộ(asynchronous communication) | 16 |
| 1. Publish/Subscribe Messaging:..... | 16 |
| 2. Event-driven Communication..... | 17 |
| Câu 5. Các dạng communication giữa các service với code cho hệ ecomSys | 19 |
| Thực hiện giữa Cart với Book, Clothes, Mobile: | 19 |
| Các function gọi API đến BookService và nhận kết quả dưới dạng Json..... | 19 |
| Các function gọi API đến MobileService và nhận kết quả dưới dạng Json | 20 |
| Các function gọi API đến ClothesService và nhận kết quả dưới dạng Json..... | 20 |
| CartService gọi các hàm trên để lấy dữ liệu từ Products | 20 |
| Demo Kết quả | 23 |
| Thực hiện giữa Search với Book, Clothes, Mobile:..... | 25 |
| Các function gọi API đến BookService, MobileService, ClothesService và nhận kết quả dưới dạng Json | 25 |
| Tạo API cho BookSearch | 26 |
| Tìm kiếm book theo name, tác giả, nhà xuất bản và danh mục | 27 |
| Demo kết quả | 27 |
| 6. Xây dựng biểu đồ activity cho tương tác giữa các services | 28 |
| 7. Xây dựng Biểu đồ lớp Phân tích cho từng service riêng lẻ | 28 |
| 1. UserService | 28 |
| 2. CartService..... | 29 |
| 3. OrderService | 29 |
| 4. BookService | 29 |
| 5. MobileService | 29 |
| 6. ClothesService | 30 |
| 7. ShipmentService | 30 |
| 8. PaymentService..... | 30 |
| 9. SearchService | 30 |
| 8. Xây dựng data model cho từng service và công nghệ phát triển tương ứng (sử dụng cả 3 mySQL, PostgreSQL, mongoDB) | 31 |
| 1. CartService..... | 31 |
| 2. UserService | 32 |
| 3. OrderService | 34 |
| 4. BookService | 35 |
| 5. MobileService | 37 |
| 6. ClothesService | 39 |
| 7. ShipmentService | 41 |
| 8. PaymentService..... | 42 |

| | |
|---|----|
| 9. SearchService | 43 |
| 9. Xây dựng biểu đồ lớp thiết kế và kiến trúc cho từng service với MVT Django | 44 |
| 10. Code và demo | 44 |
| 1. CartService..... | 44 |
| 2. UserService | 45 |
| 3. OrderService | 46 |
| 4. BookService | 48 |
| 5. MobileService | 48 |
| 6. ClothesService | 49 |
| 7. ShipmentService | 50 |
| 8. PaymentService..... | 50 |
| 9. SearchService..... | 51 |
| 10. Demo..... | 52 |

Câu 1: Mô tả bằng dạng Bảng và ngôn ngữ tự nhiên các chức năng tương ứng với các actor và phi chức năng của Hệ thống ecomSys

Các actor chính:

- Admin(Quản lý)
- Customer(Khách hàng)

Các actor phụ:

- Shipper (Người giao hàng)
- Supplier(Nhà cung cấp)
- Bank(Ngân hàng)

1. Mô tả ngôn ngữ tự nhiên

Customer Module

- Customer có thể đăng ký/đăng nhập/đăng xuất tài khoản hệ thống
- Customer có thể view details thông tin của bản thân
- Customer có thể edit thông tin của bản thân
- Customer có thể đổi mật khẩu của bản thân

Manager Module

- Admin có thể view details thông tin của bản thân
- Admin có thể edit thông tin của bản thân
- Admin có thể đổi mật khẩu của bản thân
- Admin có thể quản lý Customers: view details, add, edit, delete.
- Admin có thể xem thống kê products được mua nhiều nhất/lợi nhuận cao nhất
- Admin có thể xem thống kê Customer theo số lượng products/tổng tiền

Book Module

- Admin có thể view details các Book
- Admin có thể add một Book
- Admin có thể edit/delete một Book
- Customer có thể view list/details Book

Mobile Module

- Admin có thể view details các Mobile
- Admin có thể add một Mobile
- Admin có thể edit/delete một Mobile
- Customer có thể view list/details Mobile

Clothes Module

- Admin có thể view details các Clothes
- Admin có thể add một Clothes
- Admin có thể edit/delete một Clothes
- Customer có thể view list/details Clothes

Search Module

- Admin và Customer có thể tìm kiếm sản phẩm (Book, Clothes, Mobile) bằng keyword, voice.
- Admin có thể tìm kiếm đơn hàng theo mã Customer

- Admin và Customer có thể tìm kiếm đơn hàng theo mã Order

Cart Module

- Customer có thể add product to Cart
- Customer có thể edit/delete product trong Cart

Order Module

- Admin có thể duyệt (thêm) đơn hàng mới
- Admin có thể view list/details đơn hàng
- Admin có thể edit đơn hàng
- Admin có thể cancel đơn hàng
- Customer có thể đặt hàng
- Customer có thể view list/details thông tin đơn hàng của bản thân
- Customer có thể hủy đơn hàng nếu đơn hàng chưa được vận chuyển

Payment Module

- Bank có thể xử lý thanh toán
- Customer có thể xem các phương thức thanh toán
- Customer có thể view history thanh toán
- Admin có thể xem lịch sử giao dịch

Shipment Module

- Admin có thể add thông tin về Người giao hàng
- Admin có thể edit/delete thông tin về Người giao hàng
- Admin có thể view list/details về Người giao hàng
- Customer và Shipper có thể view details thông tin vận chuyển

2. Mô tả bằng dạng Bảng

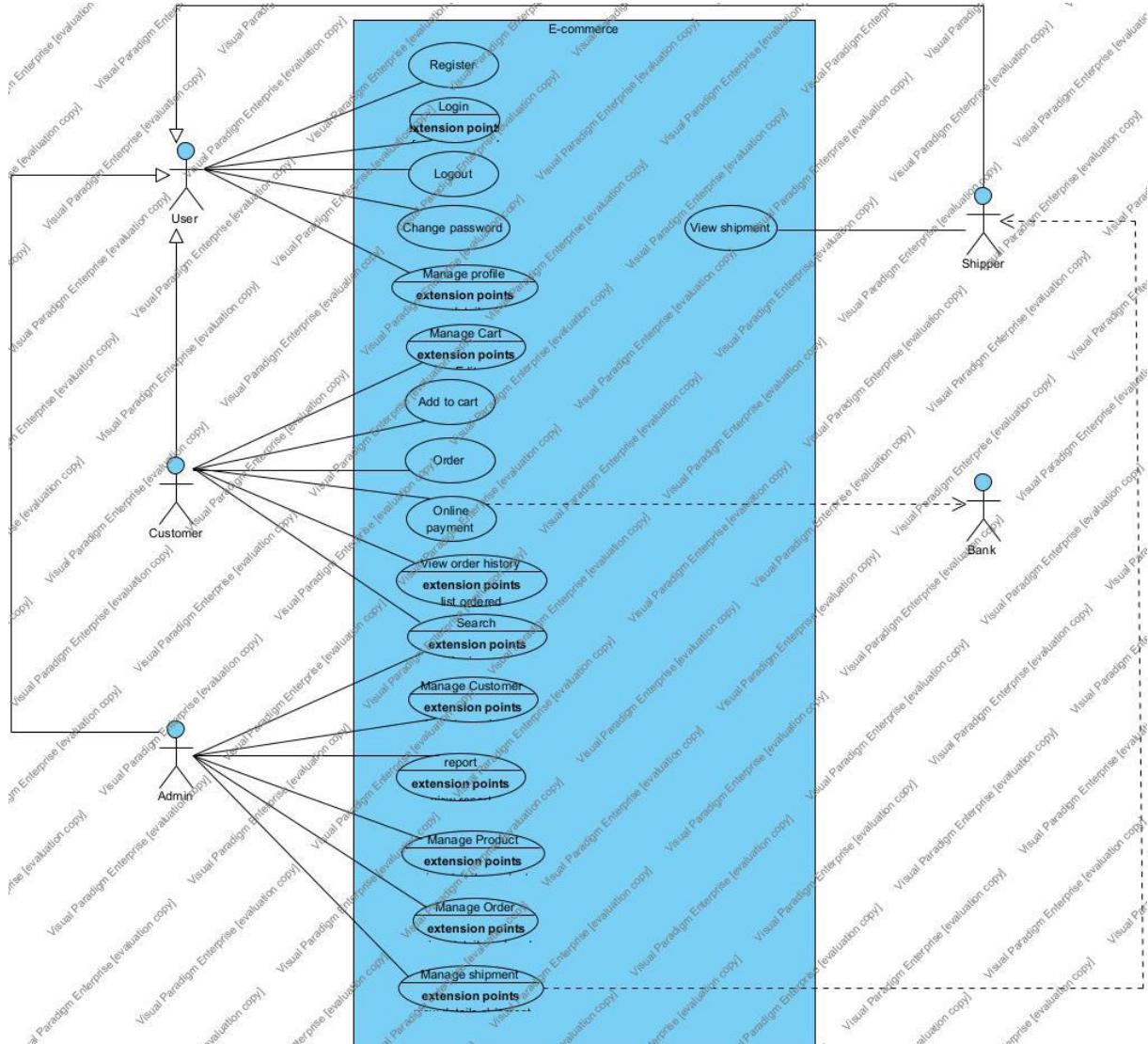
| Actor | Functionalities | Detailed Description |
|----------|-----------------|--|
| Customer | Register | Người dùng có thể đăng ký tài khoản để đăng nhập vào hệ thống |
| | Login | Người dùng sử dụng tài khoản cá nhân để đăng nhập vào hệ thống qua username, password |
| | Logout | Người dùng có thể đăng xuất khỏi hệ thống |
| | Search | <ul style="list-style-type: none"> • By Brower: Người dùng chọn mặt hàng trên web để xem chi tiết sản phẩm • By word: Người dùng nhập text vào ô tìm kiếm để xem sản phẩm • By voice: Người dùng sử dụng giọng nói để tìm kiếm sản phẩm • Order: Người dùng nhập mã Order vào ô tìm kiếm |
| | Manage Cart | <ul style="list-style-type: none"> • Edit: Người dùng vào giỏ hàng nhập số lượng sản phẩm • Delete: người dùng xóa sản phẩm ra khỏi giỏ hàng của bản thân |
| | Add to cart | Người dùng thêm sản phẩm vào giỏ hàng |
| | Order | Người dùng order các sản phẩm trong giỏ hàng |
| | Pay | Người dùng thanh toán hóa đơn mua hàng |
| | Manage profile | <ul style="list-style-type: none"> • View profile: Người dùng xem profile cá nhân của bản thân trên hệ thống |

| | | |
|-------|--------------------|---|
| | | <ul style="list-style-type: none"> Update profile: Người dùng thay đổi các thông tin cá nhân của bản thân trên hệ thống |
| | Change password | Người dùng thay đổi mật khẩu của tài khoản cá nhân |
| | View order history | Người dùng có thể xem lại lịch sử đặt hàng của bản thân trên hệ thống |
| Admin | Login | Admin sử dụng tài khoản cá nhân để đăng nhập vào hệ thống qua username, password |
| | Logout | Admin có thể đăng xuất khỏi hệ thống |
| | Manage profile | <ul style="list-style-type: none"> View profile: Admin xem profile cá nhân của bản thân trên hệ thống Update profile: Admin thay đổi các thông tin cá nhân của bản thân trên hệ thống |
| | Change password | Người dùng thay đổi mật khẩu của tài khoản cá nhân |
| | Manage Customer | <ul style="list-style-type: none"> Add customer: admin thêm người dùng vào hệ thống Edit customer: admin thay đổi thông tin của người dùng View List/detail of customer: admin xem thông tin của người dùng Delete customer: admin xoá tài khoản customer khỏi hệ thống |
| | Thống kê products | Admin xem thống kê products được mua nhiều nhất hoặc lợi nhuận cao nhất |
| | Thống kê Customer | Admin có thể xem thống kê Customer theo số lượng products hoặc tổng tiền |
| | Manage Product | <ul style="list-style-type: none"> Add product: admin thêm product vào hệ thống Edit product: admin thay đổi thông tin product View detail of product: admin xem thông tin của product Delete product: admin xoá product khỏi hệ thống |
| | Search | <ul style="list-style-type: none"> Customer: Admin nhập mã Customer vào ô tìm kiếm Order: Admin nhập mã Order vào ô tìm kiếm By Brower: Admin chọn mặt hàng trên web để xem chi tiết sản phẩm By word: Admin nhập text vào ô tìm kiếm để xem sản phẩm By voice: Admin sử dụng giọng nói để tìm kiếm sản phẩm |
| | Manage Order | <ul style="list-style-type: none"> Add order: admin thêm order vào hệ thống Edit order: admin thay đổi thông tin order View detail of order: admin xem thông tin của order Cancel order: admin hủy order khỏi hệ thống |
| | Manage shipment | <ul style="list-style-type: none"> Add shipment: admin thêm shipment vào hệ thống Edit shipment: admin thay đổi thông tin shipment View details shipment: admin xem thông tin của shipment delete shipment: admin xóa shipment khỏi hệ thống |

| | | |
|---------|----------------|--|
| Bank | Online payment | Cho phép người dùng thanh toán hóa đơn trên hệ thống mua sắm bằng tiền trong tài khoản ngân hàng, ví điện tử |
| Shipper | Register | Đăng ký vào đội shipper của hệ thống |
| | View shipment | Xem chi tiết thông tin vận chuyển |

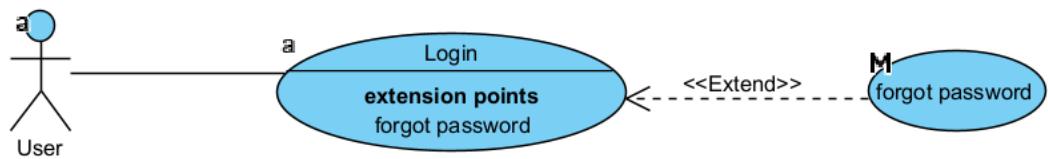
Câu 2. Vẽ biểu đồ use case tổng quát và biểu đồ use case chi tiết cho từng chức năng dịch vụ

Usecase tổng quát:

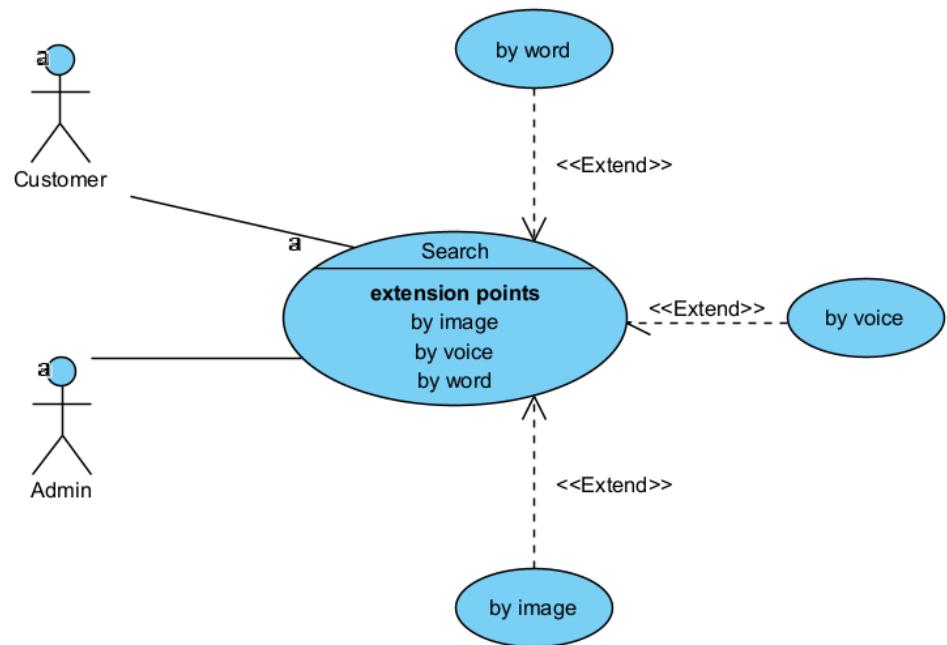


Usecase chi tiết:

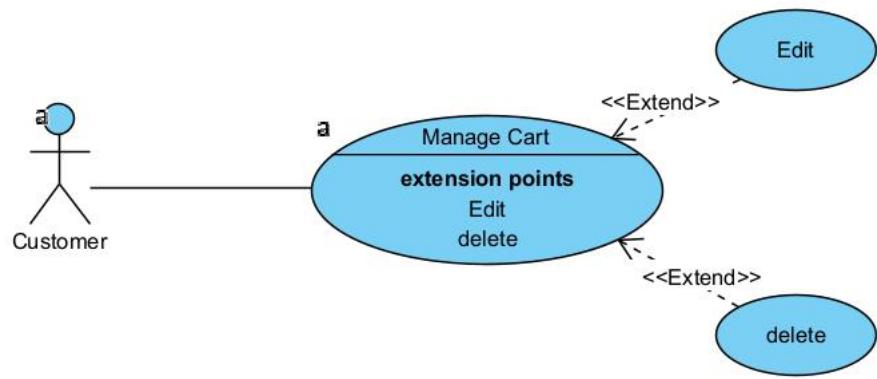
Usecase Login:



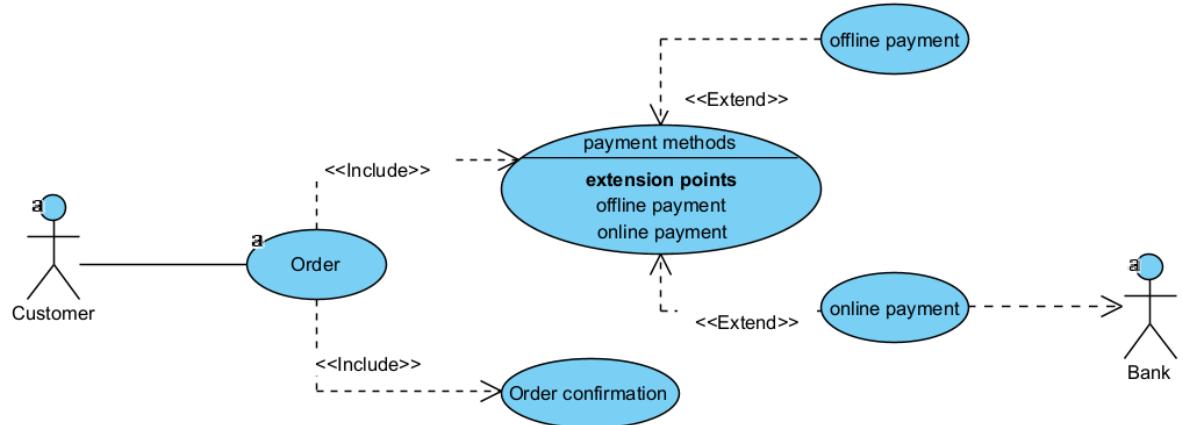
Usecase Search



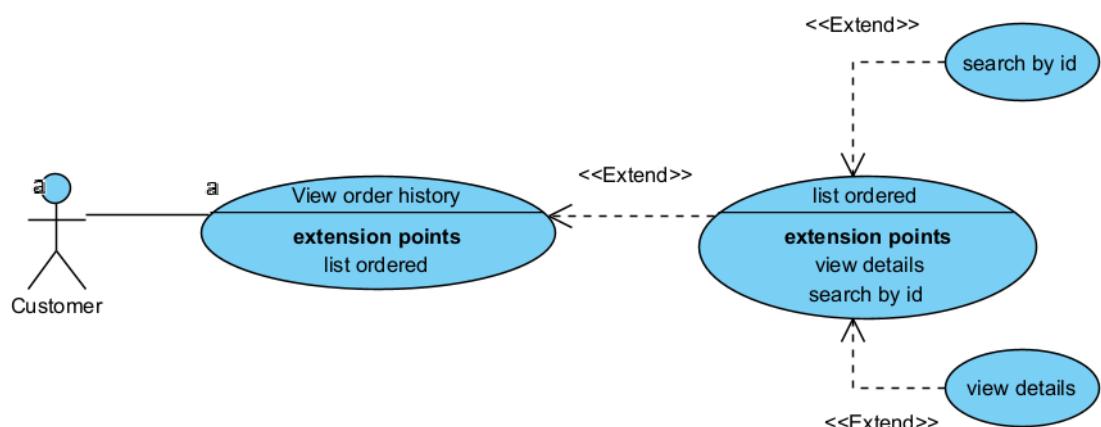
Usecase Manage Cart



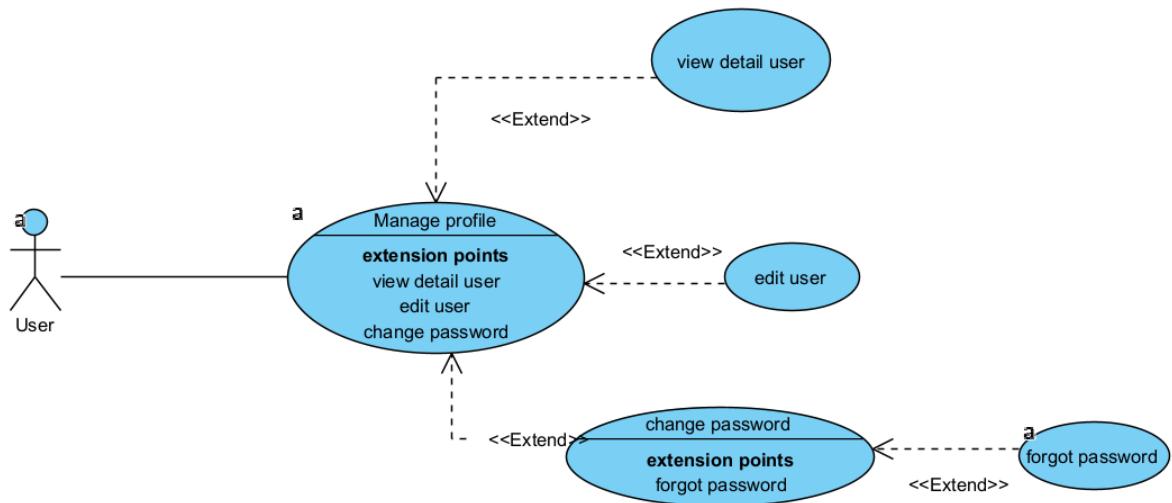
Usecase Order and Payment



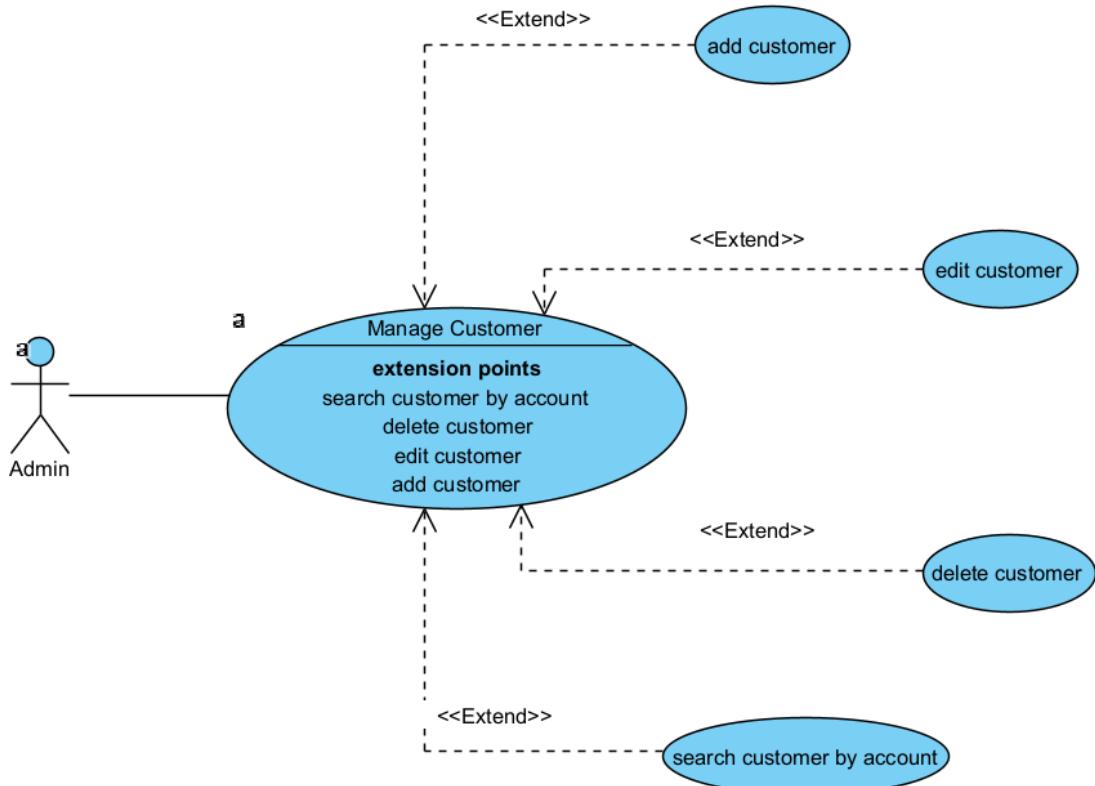
Usecase Customer history ordered



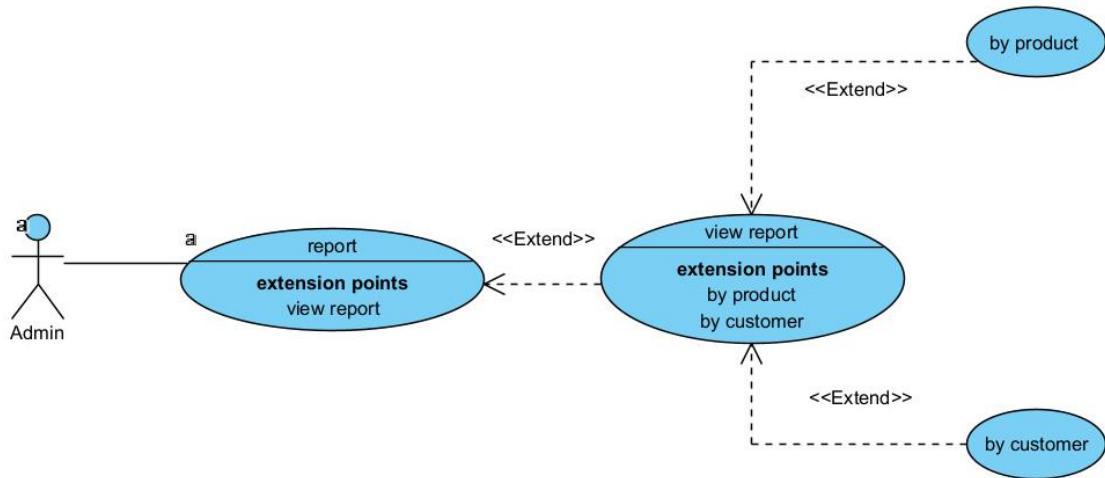
Usecase Manage profile



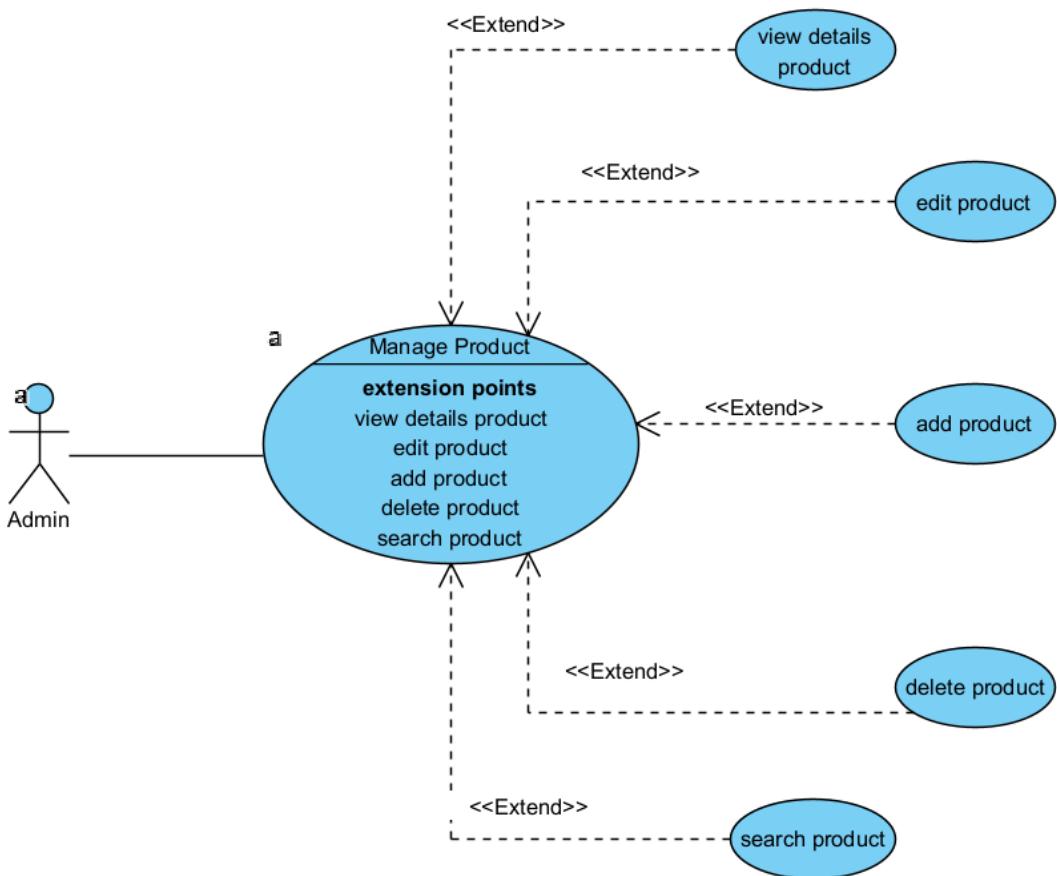
Usecase Manage Customer



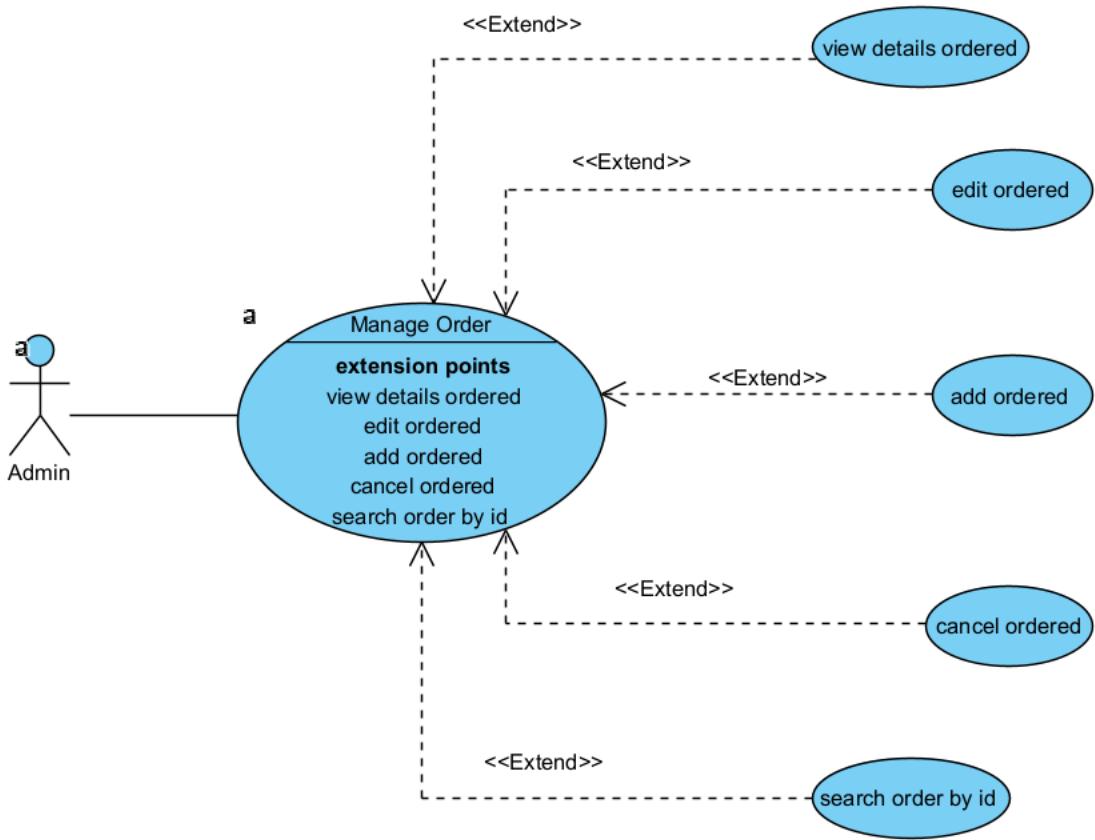
Usecase Report



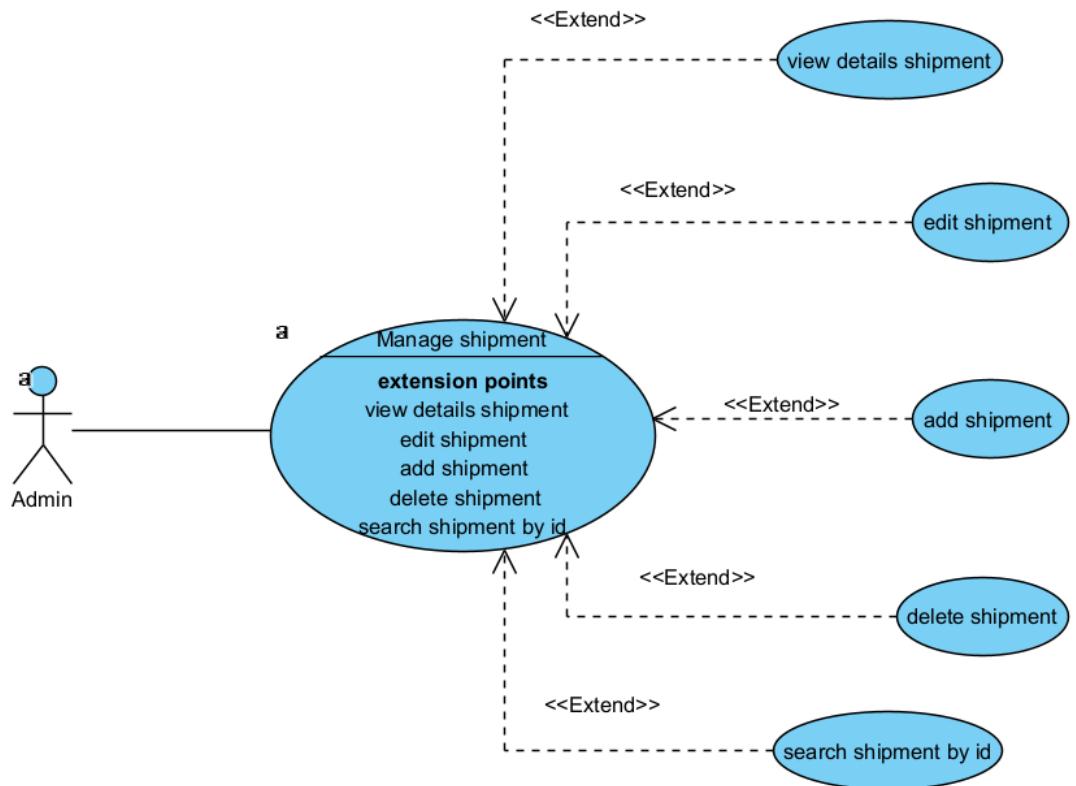
Usecase Manage Product



Usecase Manage Order



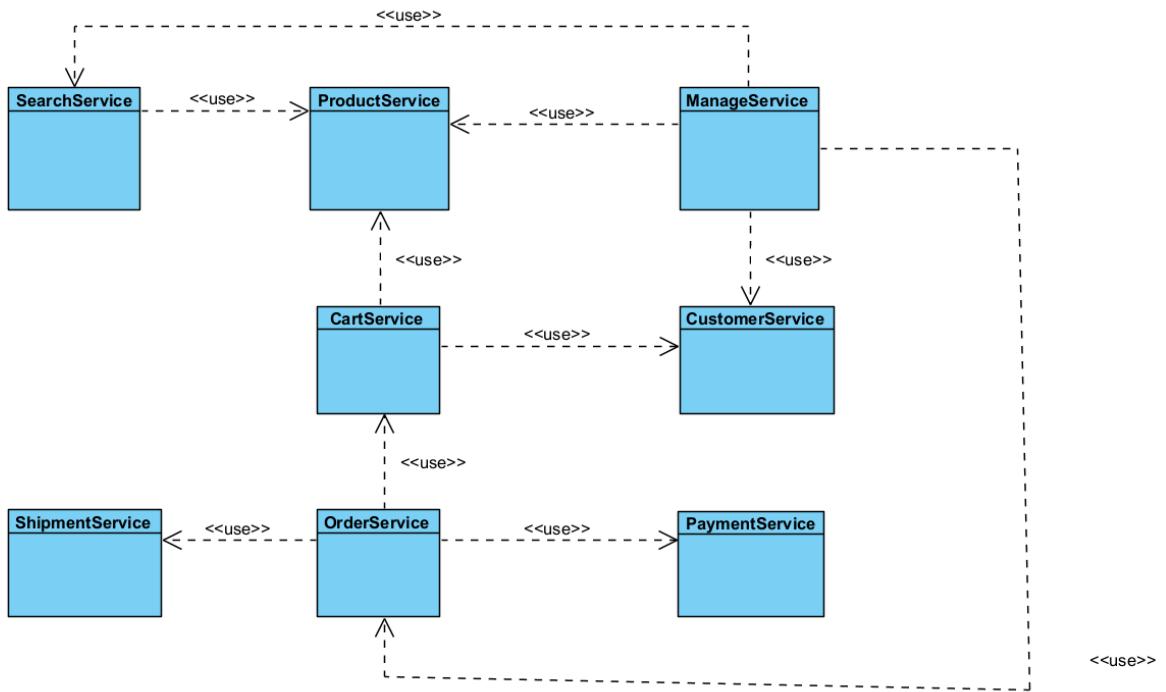
Usecase Manage shipment



Câu 3 Vẽ biểu đồ phân rã Hệ ecomSys thành các service và các tương tác giữa các dịch vụ (sử dụng quan hệ << use >> trong UML).

Phân rã dịch vụ như sau:

1. CustomerService: Dịch vụ này liên quan đến việc quản lý tài khoản người dùng và xác thực, chẳng hạn như cho phép người dùng tạo và đăng nhập vào tài khoản của họ cũng như lưu trữ thông tin hồ sơ người dùng.
2. ProductService: Dịch vụ này liên quan đến việc quản lý sản phẩm có sẵn để bán trên trang web, chẳng hạn như thêm sản phẩm mới, cập nhật sản phẩm, xoá sản phẩm hay xem chi tiết sản phẩm.
3. SearchService: Dịch vụ này liên quan đến việc tìm kiếm các sản phẩm trên trang web để xem chi tiết, thêm giỏ hàng hoặc mua sắm. Có thể tìm kiếm bằng text, voice, image.
4. CartService: Dịch vụ này có khả năng chịu trách nhiệm quản lý các chức năng giỏ hàng của trang web, chẳng hạn như cho phép người dùng thêm các mặt hàng vào giỏ hàng của họ, cập nhật giỏ hàng khi các mặt hàng được thêm vào hoặc xoá và tính toán tổng chi phí của đơn hàng.
5. OrderService: Dịch vụ này liên quan đến quản lý quy trình đặt hàng và thực hiện đơn hàng, chẳng hạn như nhận và xử lý đơn hàng, theo dõi mức tồn kho và cập nhật cho khách hàng về trạng thái đơn hàng.
6. PaymentService: Dịch vụ này sẽ xử lý quá trình thanh toán, chẳng hạn như chấp nhận thanh toán bằng tài khoản online (ví điện tử), xác minh tính hợp lệ của thông tin thanh toán và bắt đầu giao dịch với cổng thanh toán hoặc ngân hàng.
7. ShipmentService: Dịch vụ này sẽ quản lý việc vận chuyển và giao sản phẩm cho khách hàng, chẳng hạn như theo dõi các gói hàng, phối hợp với người vận chuyển và tính toán chi phí vận chuyển.
8. ManageService: Dịch vụ này có thể liên quan đến việc quản lý tài khoản và quyền của người dùng, chẳng hạn như cho phép người dùng truy cập vào một số phần nhất định của trang web và theo dõi chỉ số hiệu suất của nhân viên



Câu 4 Trình bày các dạng communication giữa các service với nhau (synchronous và asynchronous) với code và ví dụ

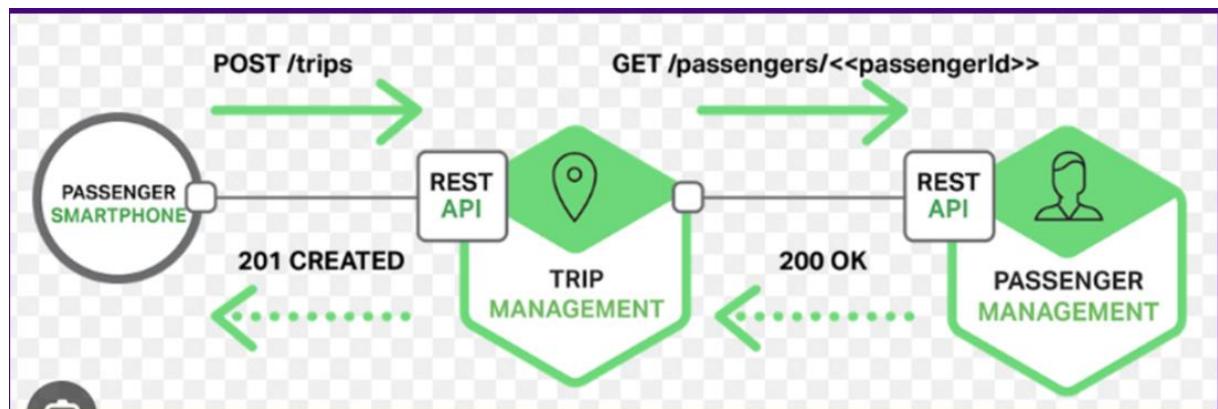
Các dạng communication giữa các dịch vụ có thể được chia thành hai loại chính: đồng bộ (synchronous) và không đồng bộ (asynchronous). Dưới đây là một trình bày về cả hai loại:

Giao tiếp Đồng bộ (Synchronous Communication):

Trong giao tiếp đồng bộ, máy khách gửi yêu cầu đến microservice và chờ phản hồi trước khi tiếp tục. Tuy nhiên, nó có thể dẫn đến tắc nghẽn về hiệu suất và khả năng xảy ra lỗi xếp tầng nếu một vi dịch vụ gặp phải thời gian ngừng hoạt động.

1. HTTP/HTTPS Request-Response:

Các microservice giao tiếp đồng bộ bằng giao thức HTTP hoặc HTTPS, trong đó máy khách gửi yêu cầu HTTP đến máy chủ HTTP và chờ phản hồi của nó.



Ưu điểm:

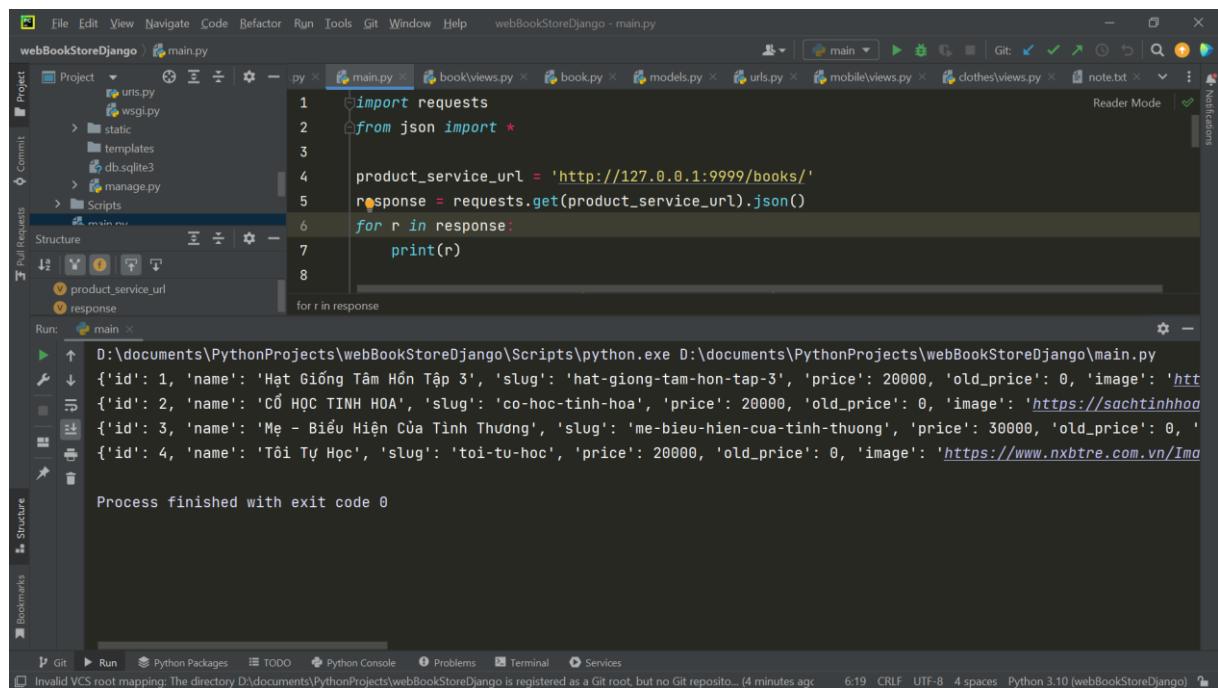
- Giải pháp đơn giản xảy ra trong thời gian thực

Nhược điểm

- Máy khách chặn luồng của nó khi nó gọi máy chủ, chỉ tiếp tục nhiệm vụ khi có phản hồi từ máy chủ
- Có thể phải xử lý lỗi xếp tầng

Ví dụ: Mô hình Request-Response trong HTTP, trong đó một client gửi yêu cầu HTTP đến server và đợi cho đến khi server trả về phản hồi.

Dưới đây gửi một request đến url: <http://127.0.0.1:9999/books/> để nhận Response kết quả xác thực cho việc lấy tất cả book



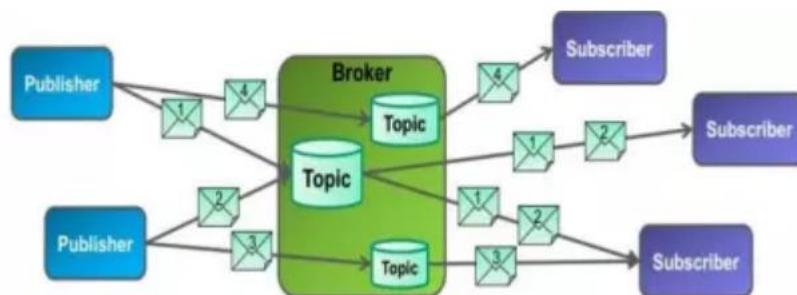
```
File Edit View Navigate Code Refactor Run Tools Git Window Help webBookStoreDjango - main.py
main.py
1 import requests
2 from json import *
3
4 product_service_url = 'http://127.0.0.1:9999/books/'
5 response = requests.get(product_service_url).json()
6 for r in response:
7     print(r)
8
for r in response:
    print(r)

Run: main x
D:\documents\PythonProjects\webBookStoreDjango\Scripts\python.exe D:\documents\PythonProjects\webBookStoreDjango\main.py
{'id': 1, 'name': 'Hạt Giống Tâm Hồn Tập 3', 'slug': 'hat-giong-tam-hon-tap-3', 'price': 20000, 'old_price': 0, 'image': 'http://www.nxbtre.com.vn/Ima
{'id': 2, 'name': 'Cố HỌC TINH HOA', 'slug': 'co-hoc-tinh-hoa', 'price': 20000, 'old_price': 0, 'image': 'https://sachtinhhoa
{'id': 3, 'name': 'Mẹ - Biểu Hiện Của Tình Thương', 'slug': 'me-bieu-hien-cua-tinh-thuong', 'price': 30000, 'old_price': 0, 'image': 'http://www.nxbtre.com.vn/Ima
{'id': 4, 'name': 'Tôi Tự Học', 'slug': 'toi-tu-hoc', 'price': 20000, 'old_price': 0, 'image': 'https://www.nxbtre.com.vn/Ima
Process finished with exit code 0
```

Giao Tiếp Không Đồng Bộ(asynchronous communication)

Giao tiếp không đồng bộ cho phép client gửi tin nhắn đến một microservice mà không cần chờ phản hồi ngay lập tức. Tuy nhiên, nó gây ra sự phức tạp trong việc xử lý tính nhất quán và thông báo cuối cùng.

1. Publish/Subscribe Messaging:



Ưu điểm:

- Message xuất bản/đăng ký cung cấp khả năng liên lạc theo thời gian thực vì nó gửi message ngay lập tức đến người đăng ký.
- Nó mang lại khả năng mở rộng cao hơn vì không có số lượng nhà xuất bản và người đăng ký được xác định trước
- chúng có thể được thêm hoặc xóa bất cứ lúc nào

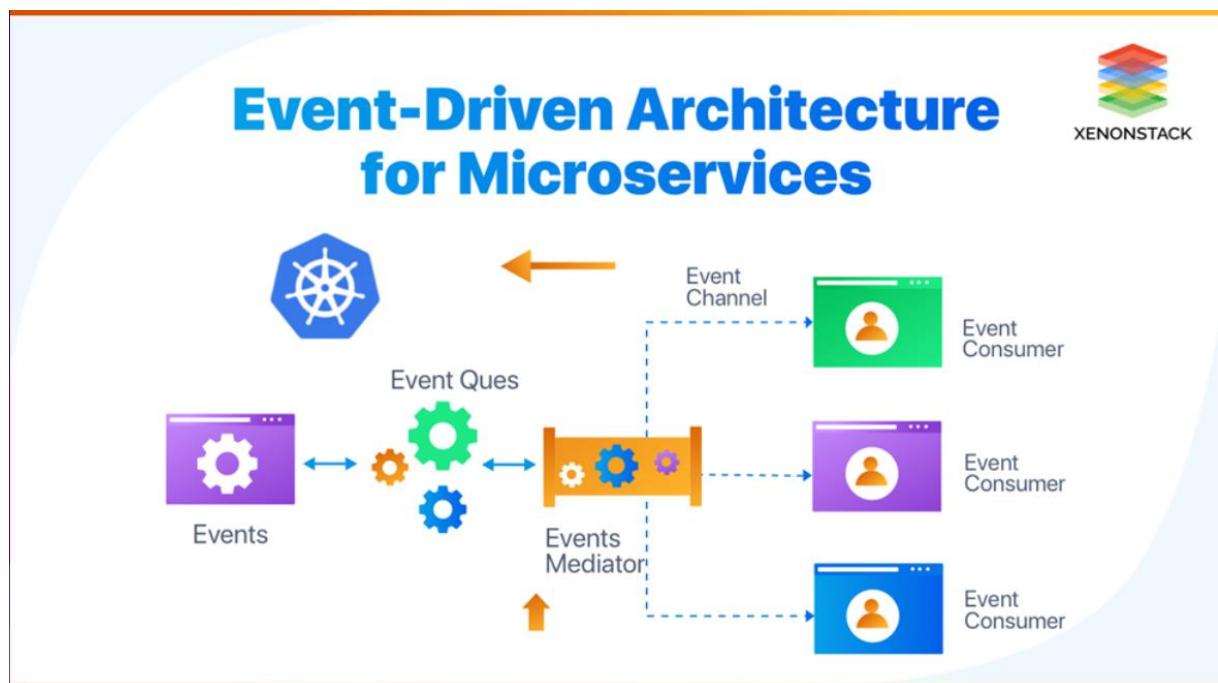
Nhược điểm:

- Máy chủ môi giới (Broker) không cần thông báo về trạng thái gửi thông điệp. Do đó không có cách nào để phát hiện xem thông điệp đã gửi đúng hay chưa.
- Publisher không hề biết gì về trạng thái của subscribe và ngược lại. Vậy làm sao chúng ta có thể đảm bảo mọi thứ đều ổn.
- Những kẻ xấu (Malicious Publisher) có thể gửi những thông điệp xấu, và các Subscriber sẽ truy cập vào những thứ mà họ không nhận.

Link tham khảo: <https://viblo.asia/p/mqtt-la-gi-vai-tro-cua-mqtt-trong-iot-V3m5WL3bKO7>

2. Event-driven Communication

Microservice sẽ phát ra các sự kiện khi một số hành động nhất định xảy ra. Các dịch vụ khác có thể đăng ký các sự kiện này và phản hồi tương ứng



Ưu điểm

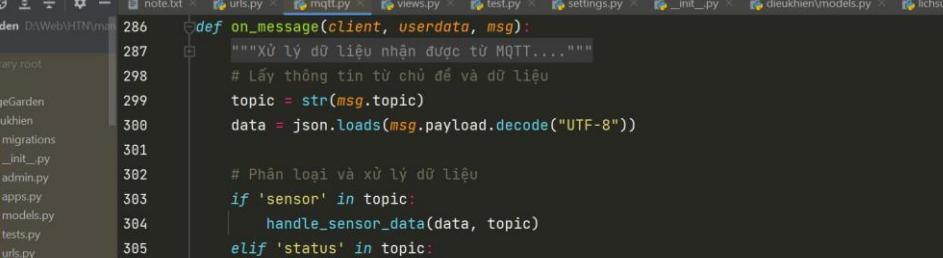
- Truyền phát sự kiện cung cấp khả năng tương tác theo thời gian thực để nhận thức dữ liệu tốt hơn. Nó sử dụng các sự kiện bắt biến và thứ tự thông điệp, cung cấp tính đồng thời trong các hệ thống phân tán.
- Các luồng cung cấp sự ghép nối lỏng lẻo giữa các dịch vụ; các dịch vụ tạo ra sự kiện không cần biết các sự kiện được tiêu thụ như thế nào và các dịch vụ tiêu thụ các sự kiện không cần biết các sự kiện được tạo ra như thế nào.

- Consumer cũng có được khả năng chịu lỗi và khả năng phục hồi vì nếu consumer gặp lỗi, hệ thống sẽ tiếp tục hoạt động khi các message được xếp hàng đợi trong trình message broker; consumer có thể tiếp tục sử dụng các sự kiện sau khi nó phục hồi sau thất bại.

Nhược điểm

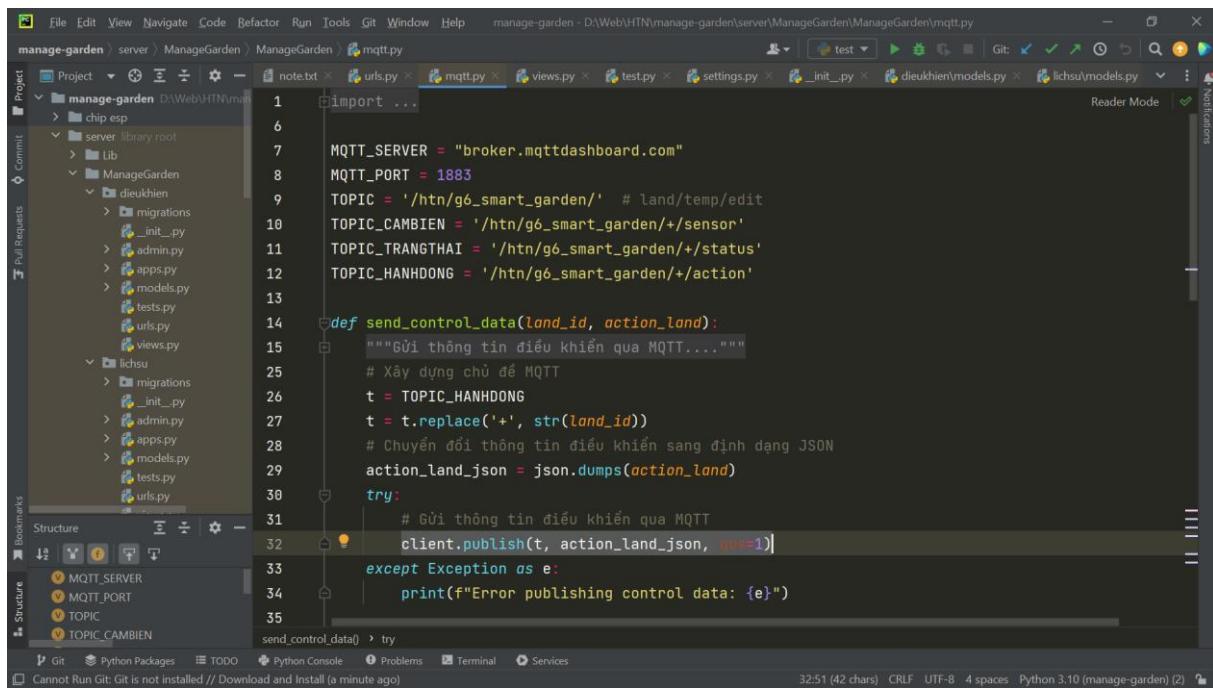
- Truyền phát sự kiện dẫn đến độ phức tạp của hệ thống tăng lên, đòi hỏi hệ thống xếp hàng với publisher và consumer ở cả hai đầu.
 - Điều này dẫn đến chi phí tăng lên, vì ngoài logic nghiệp vụ, hệ thống phải đọc và xác thực các sự kiện

Ví dụ: Publish/Subscribe Messaging, trong đó các dịch vụ gửi và nhận thông điệp mà không cần phải chờ đợi phản hồi trực tiếp từ dịch vụ khác bằng giao thức MQTT



The screenshot shows the PyCharm IDE interface with the following details:

- Project:** manage-garden
- File Structure:** The project structure is visible on the left, showing the `manage-garden` directory containing `chip esp`, `server`, `library root`, and `ManageGarden`. `ManageGarden` contains `dieukhien`, `ldhsu`, and `urls.py`. `dieukhien` contains `migrations`, `__init__.py`, `admin.py`, `apps.py`, `models.py`, `tests.py`, `urls.py`, and `views.py`. `ldhsu` contains `migrations`, `__init__.py`, `admin.py`, `apps.py`, `models.py`, `tests.py`, and `urls.py`.
- Code Editor:** The main window displays the `mqtt.py` file. The code is a Python script for an MQTT client. It defines a function `on_message` that handles messages from the broker. It then creates a `paho.Client` object, sets its callbacks, and connects to the MQTT broker. It also subscribes to two topics: `TOPIC_CAMBIEU` and `TOPIC_TRANGTHAI`, with a qos of 1.
- Toolbars and Status Bar:** The top bar shows the file menu (File, Edit, View, Navigate, Code, Refactor, Run, Tools, Git, Window, Help) and the current file path (manage-garden - D:\Web\HTN\manage-garden\server\ManageGarden\ManageGarden\mqtt.py). The bottom bar shows Git status (Cannot Run Git: Git is not installed // Download and Install (2 minutes ago)), Python version (Python 3.10 (manage-garden) (2)), and other system information (422:41 (79 chars, 1 line break) CRLF - UTF-8 4 spaces Python 3.10 (manage-garden) (2)).



```

import ...

MQTT_SERVER = "broker.mqttdashboard.com"
MQTT_PORT = 1883
TOPIC = '/htn/g6_smart_garden/' # land/temp/edit
TOPIC_CAMBIEN = '/htn/g6_smart_garden/+/sensor'
TOPIC_TRANGTHAI = '/htn/g6_smart_garden/+/status'
TOPIC_HANHDONG = '/htn/g6_smart_garden/+/action'

def send_control_data(land_id, action_land):
    """Gửi thông tin điều khiển qua MQTT...."""
    # Xây dựng chủ đề MQTT
    t = TOPIC_HANHDONG
    t = t.replace('+', str(land_id))
    # Chuyển đổi thông tin điều khiển sang định dạng JSON
    action_land_json = json.dumps(action_land)
    try:
        # Gửi thông tin điều khiển qua MQTT
        client.publish(t, action_land_json, qos=1)
    except Exception as e:
        print(f"Error publishing control data: {e}")

send_control_data() try

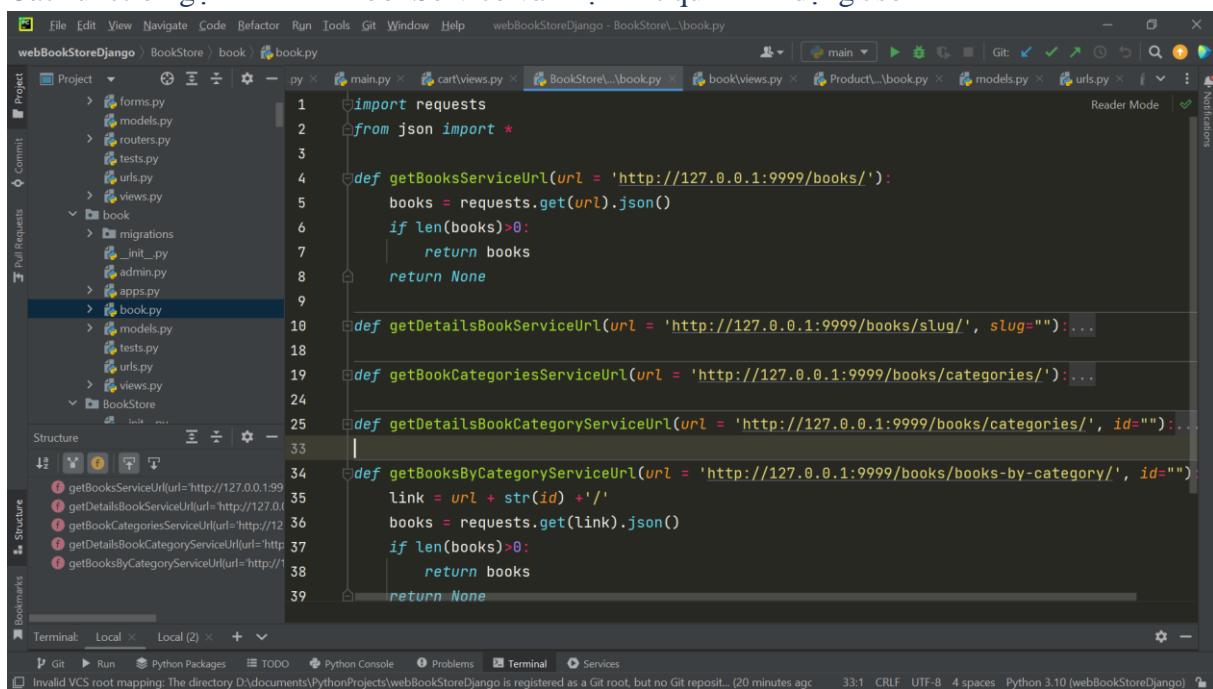
```

Câu 5. Các dạng communication giữa các service với code cho hệ ecomSys

Giao tiếp đồng bộ trong hệ ecomSys

Thực hiện giữa Cart với Book, Clothes, Mobile:

Các function gọi API đến BookService và nhận kết quả dưới dạng Json



```

import requests
from json import *

def getBooksServiceUrl(url = 'http://127.0.0.1:9999/books/'):
    books = requests.get(url).json()
    if len(books)>0:
        return books
    return None

def getDetailsBookServiceUrl(url = 'http://127.0.0.1:9999/books/slug/', slug=""):
    ...

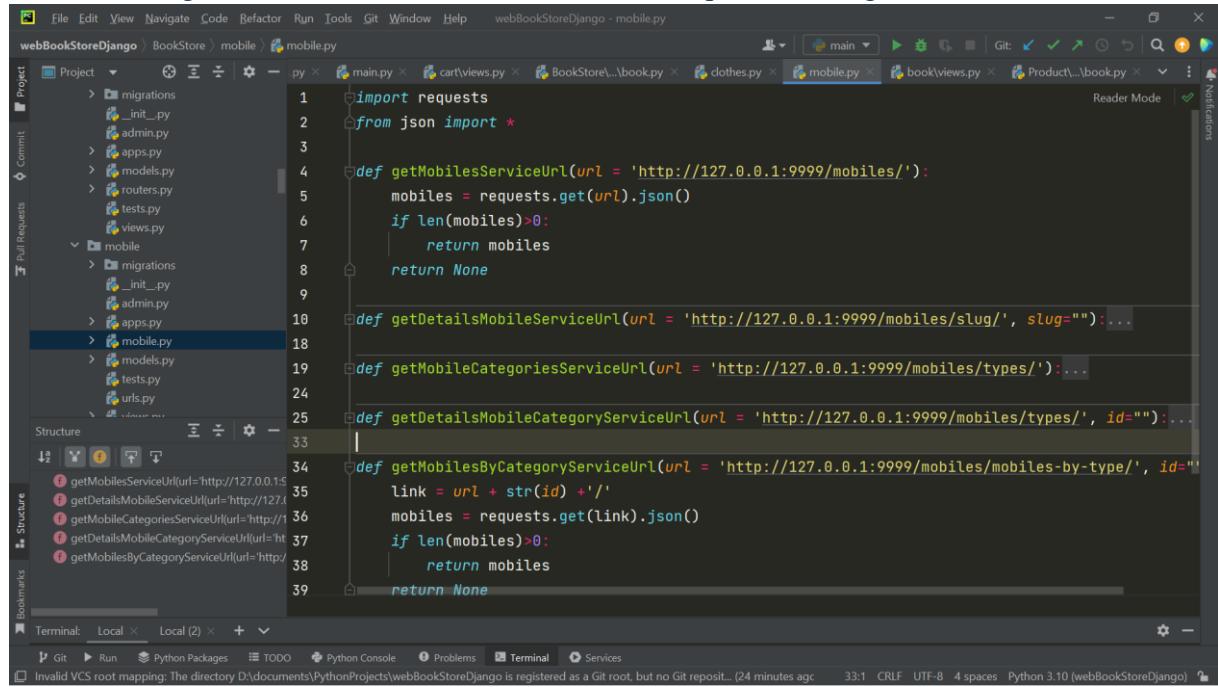
def getBookCategoriesServiceUrl(url = 'http://127.0.0.1:9999/books/categories/'):
    ...

def getDetailsBookCategoryServiceUrl(url = 'http://127.0.0.1:9999/books/categories/', id=""):
    ...

def getBooksByCategoryServiceUrl(url = 'http://127.0.0.1:9999/books/books-by-category/', id=""):
    link = url + str(id) + '/'
    books = requests.get(link).json()
    if len(books)>0:
        return books
    return None

```

Các function gọi API đến MobileService và nhận kết quả dưới dạng Json

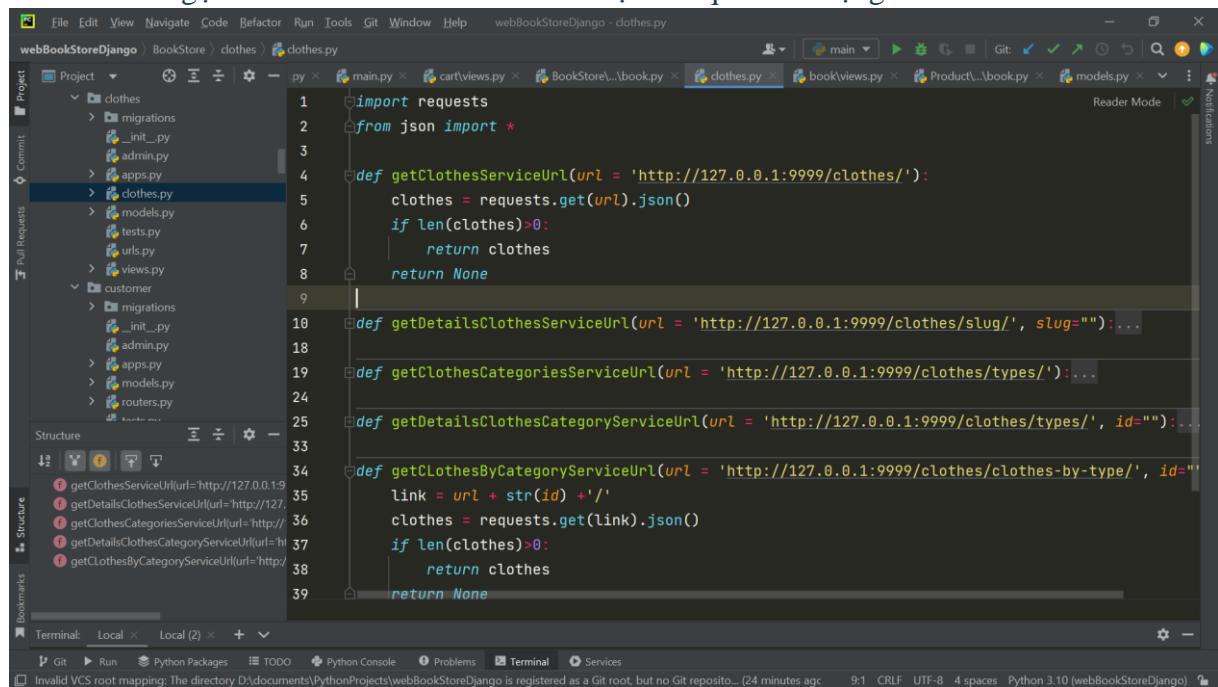


```

1  import requests
2  from json import *
3
4  def getMobilesServiceUrl(url = 'http://127.0.0.1:9999/mobiles/'):
5      mobiles = requests.get(url).json()
6      if len(mobiles)>0:
7          return mobiles
8      return None
9
10 def getDetailsMobileServiceUrl(url = 'http://127.0.0.1:9999/mobiles/slug/', slug=""):
11
12 def getMobileCategoriesServiceUrl(url = 'http://127.0.0.1:9999/mobiles/types/'):
13
14 def getDetailsMobileCategoryServiceUrl(url = 'http://127.0.0.1:9999/mobiles/types/', id=""):
15
16 def getMobilesByCategoryServiceUrl(url = 'http://127.0.0.1:9999/mobiles/mobiles-by-type/', id=""):
17     link = url + str(id) + '/'
18     mobiles = requests.get(link).json()
19     if len(mobiles)>0:
20         return mobiles
21     return None
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39

```

Các function gọi API đến ClothesService và nhận kết quả dưới dạng Json

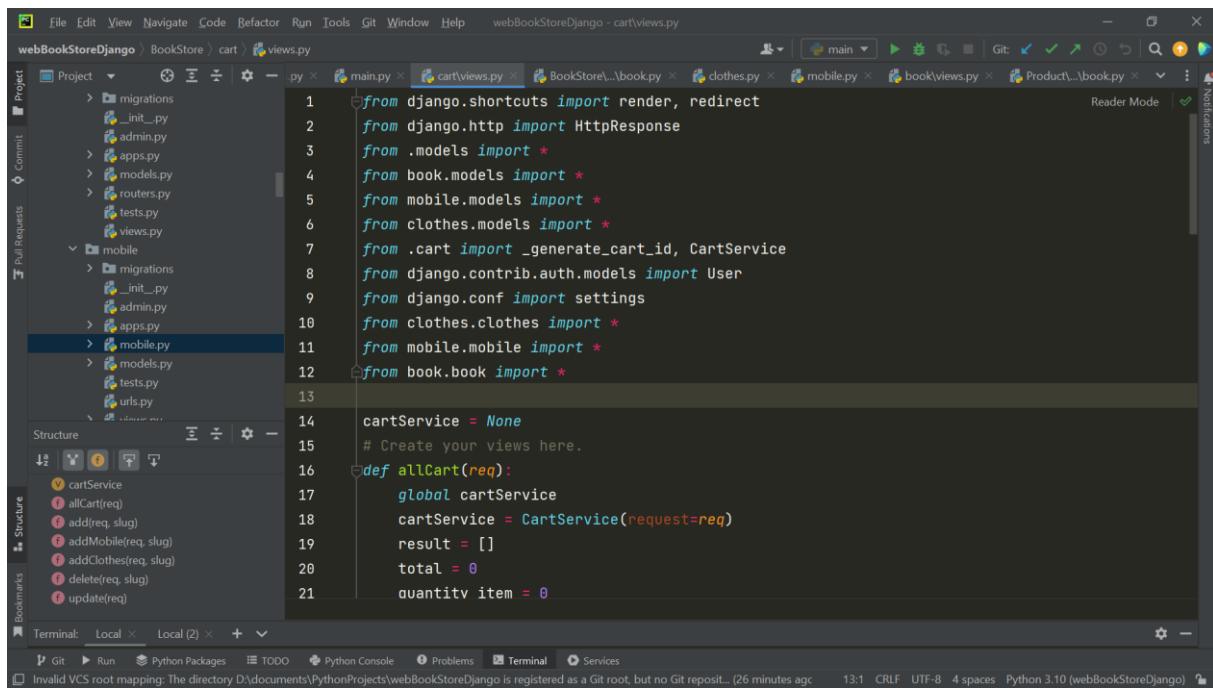


```

1  import requests
2  from json import *
3
4  def getClothesServiceUrl(url = 'http://127.0.0.1:9999/clothes/'):
5      clothes = requests.get(url).json()
6      if len(clothes)>0:
7          return clothes
8      return None
9
10 def getDetailsClothesServiceUrl(url = 'http://127.0.0.1:9999/clothes/slug/', slug=""):
11
12 def getClothesCategoriesServiceUrl(url = 'http://127.0.0.1:9999/clothes/types/'):
13
14 def getDetailsClothesCategoryServiceUrl(url = 'http://127.0.0.1:9999/clothes/types/', id=""):
15
16 def getClothesByCategoryServiceUrl(url = 'http://127.0.0.1:9999/clothes/mobiles-by-type/', id=""):
17     link = url + str(id) + '/'
18     clothes = requests.get(link).json()
19     if len(clothes)>0:
20         return clothes
21     return None
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39

```

CartService gọi các hàm trên để lấy dữ liệu từ Products
 Lấy danh sách các product trong cart



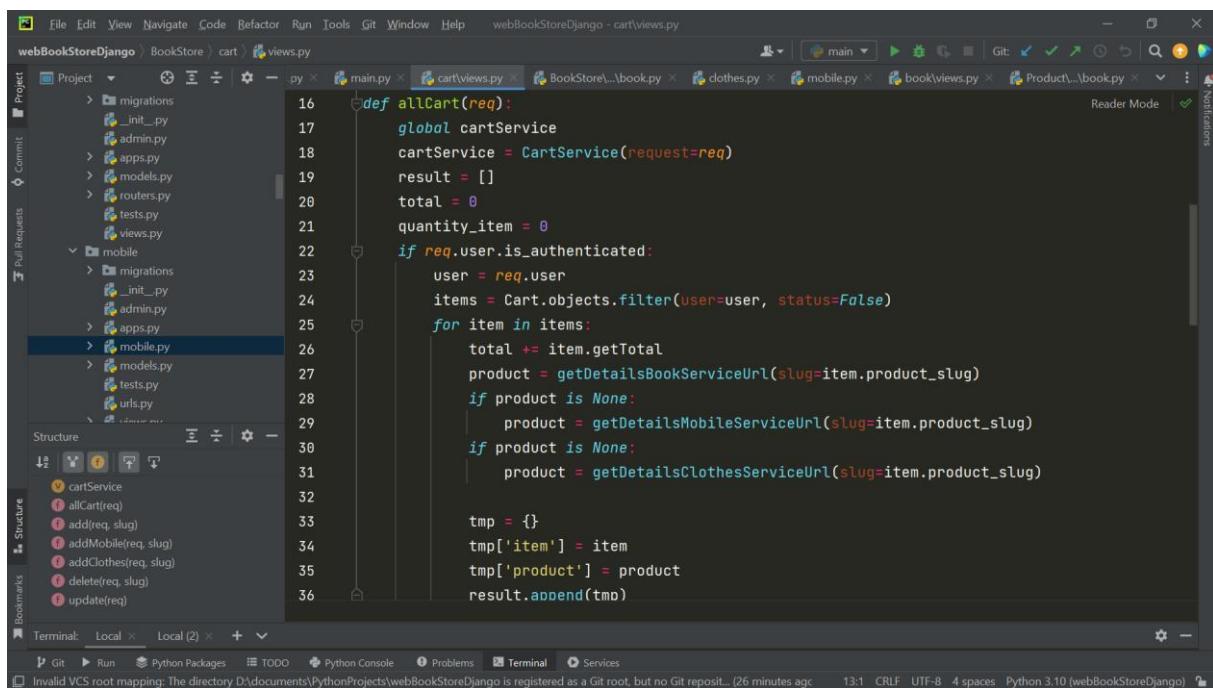
```
from django.shortcuts import render, redirect
from django.http import HttpResponseRedirect
from .models import *
from book.models import *
from mobile.models import *
from clothes.models import *
from .cart import _generate_cart_id, CartService
from django.contrib.auth.models import User
from django.conf import settings
from clothes.clothes import *
from mobile.mobile import *
from book.book import *

cartService = None
# Create your views here.
def allCart(req):
    global cartService
    cartService = CartService(request=req)
    result = []
    total = 0
    quantity_item = 0

    cartService = _generate_cart_id(req)
    user = req.user
    items = Cart.objects.filter(user=user, status=False)
    for item in items:
        total += item.getTotal()
        product = getDetailsBookServiceUrl(slug=item.product_slug)
        if product is None:
            product = getDetailsMobileServiceUrl(slug=item.product_slug)
        if product is None:
            product = getDetailsClothesServiceUrl(slug=item.product_slug)

        tmp = {}
        tmp['item'] = item
        tmp['product'] = product
        result.append(tmp)

    return render(req, 'cart/cart.html', {'result': result, 'total': total})
```



```
def allCart(req):
    global cartService
    cartService = CartService(request=req)
    result = []
    total = 0
    quantity_item = 0

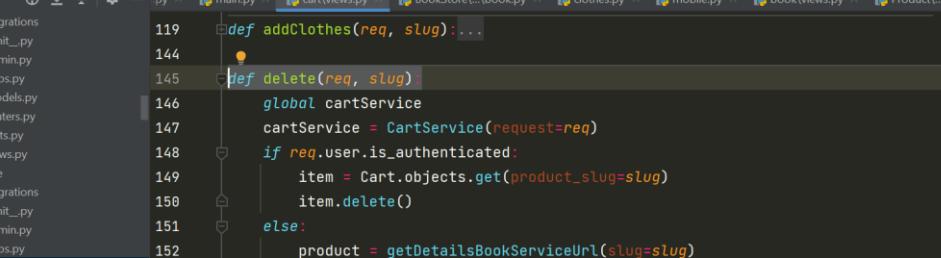
    if req.user.is_authenticated:
        user = req.user
        items = Cart.objects.filter(user=user, status=False)
        for item in items:
            total += item.getTotal()
            product = getDetailsBookServiceUrl(slug=item.product_slug)
            if product is None:
                product = getDetailsMobileServiceUrl(slug=item.product_slug)
            if product is None:
                product = getDetailsClothesServiceUrl(slug=item.product_slug)

            tmp = {}
            tmp['item'] = item
            tmp['product'] = product
            result.append(tmp)

    return render(req, 'cart/cart.html', {'result': result, 'total': total})
```

Add product to Cart

Xóa Cart



```
119     def addClothes(req, slug):...
144
145     def delete(req, slug):
146         global cartService
147         cartService = CartService(request=req)
148
149         if req.user.is_authenticated:
150             item = Cart.objects.get(product_slug=slug)
151             item.delete()
152         else:
153             product = getDetailsBookServiceUrl(slug=slug)
154             if product is None:
155                 product = getDetailsMobileServiceUrl(slug=slug)
156                 if product is None:
157                     product = getDetailsClothesServiceUrl(slug=slug)
158                     cartService.remove(product)
159
160     def update(req):...
```

Update Cart

Demo Kết quả List Product

http://127.0.0.1:9998

Microsoft OneDrive Thông báo: Chuyển... Ngôn ngữ tự nhiên... Tư Duy Kinh Doanh... suc khoe Google Xu hướng Set up your site —... thiết kế Tất cả dấu trang

Home Categories Products Account Search Items: 0

List Product!

Books

Hạt Giống Tâm Hồn Tập 3
20,000 VND

[Add Cart](#) [View](#)

CỐ HỌC TINH HOA
20,000 VND

[Add Cart](#) [View](#)

Mẹ – Biểu Hiện Của Tình Thương
30,000 VND

[Add Cart](#) [View](#)

Tôi Tự Học
20,000 VND

[Add Cart](#) [View](#)

Add vào Cart

http://127.0.0.1:9998/books/details/co-hoc-tinh-hoa

Microsoft OneDrive Thông báo: Chuyển... Ngôn ngữ tự nhiên... Tư Duy Kinh Doanh... suc khoe Google Xu hướng Set up your site —... thiết kế Tất cả dấu trang

Home Categories My Orders Products Account Search Items: 0

sachtinhhoa.vn

Name: CỐ HỌC TINH HOA
Price: 20,000 VND
Author: 2
Quantity:
Description: Co hoc tinh hoa

[Add Cart](#)

Carts

Items: 2

Total: 90,000 VND

Checkout

| Item | Price | Quantity | Total |
|--------------------------------|------------|----------|------------|
| CỎ HỌC TINH HOA | 20,000 VND | 3 | 60,000 VND |
| Mẹ – Biểu Hiện Của Tình Thương | 30,000 VND | 1 | 30,000 VND |

Thực hiện giữa Search với Book, Clothes, Mobile:

Các function gọi API đến BookService, MobileService, ClothesService và nhận kết quả dưới dạng Json

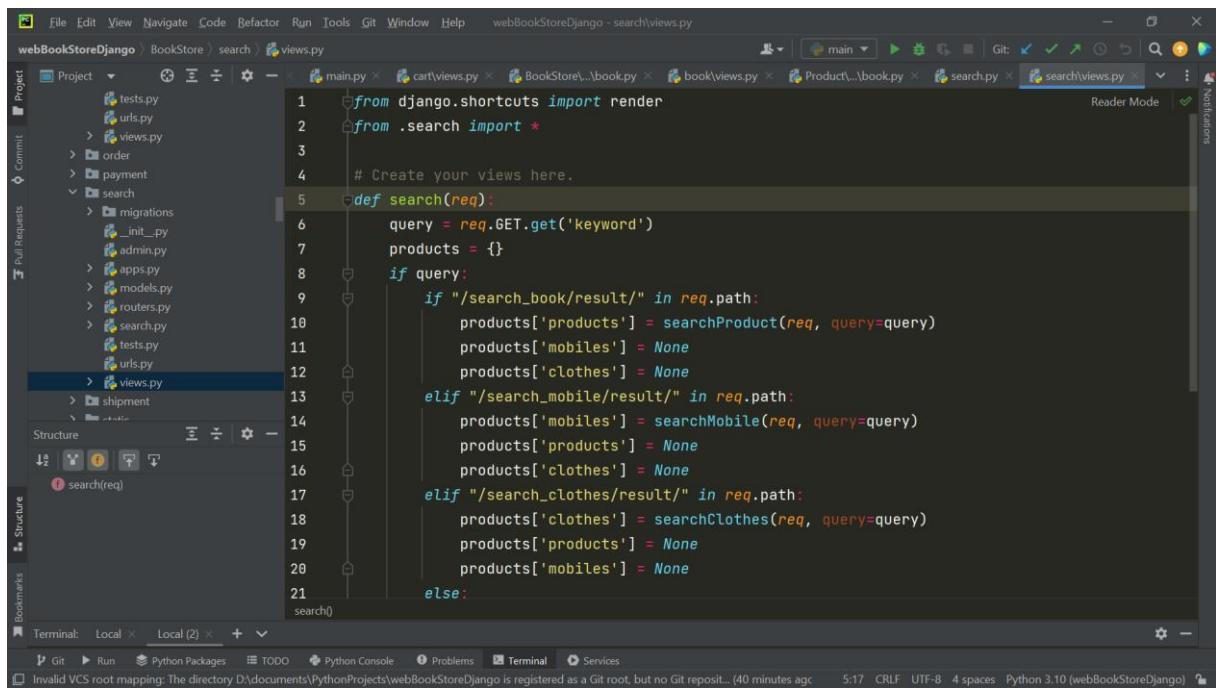
File search/search.py

```

1  from clothes.clothes import *
2  from mobile.mobile import *
3  from book.book import *
4  import requests
5  from json import *
6
7  def searchAll(req, query):
8
9      def searchProduct(req, query, url="http://127.0.0.1:9999/books/search/"):
10         products = []
11         query = str(query).lower()
12         print(query)
13         url = url + f'?query={query}'
14         products = requests.get(url).json()
15         print(products)
16         return products
17
18      def searchMobile(req, query, url="http://127.0.0.1:9999/mobiles/search/"):
19
20      def searchClothes(req, query, url="http://127.0.0.1:9999/clothes/search/"):
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

```

File search/view.py

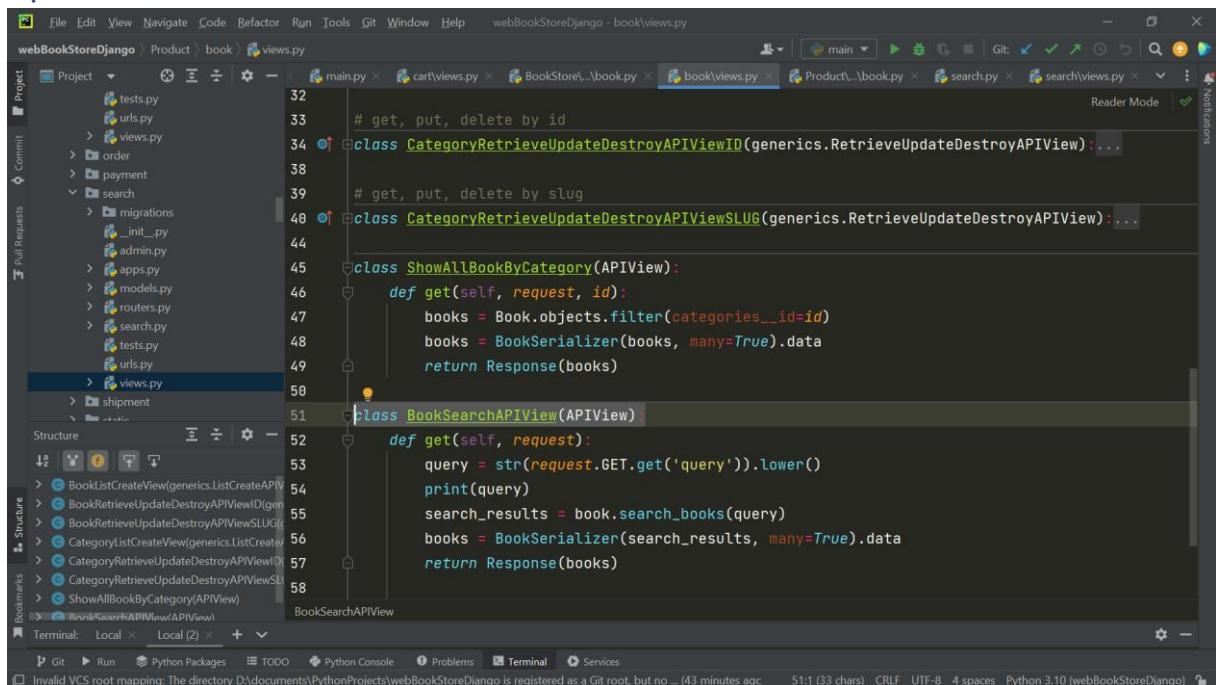


```

1  from django.shortcuts import render
2  from .search import *
3
4  # Create your views here.
5  def search(req):
6      query = req.GET.get('keyword')
7      products = {}
8
9      if query:
10         if "/search_book/result/" in req.path:
11             products['products'] = searchProduct(req, query=query)
12             products['mobiles'] = None
13             products['clothes'] = None
14         elif "/search_mobile/result/" in req.path:
15             products['mobiles'] = searchMobile(req, query=query)
16             products['products'] = None
17             products['clothes'] = None
18         elif "/search_clothes/result/" in req.path:
19             products['clothes'] = searchClothes(req, query=query)
20             products['products'] = None
21             products['mobiles'] = None
22     else:
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58

```

Tạo API cho BookSearch

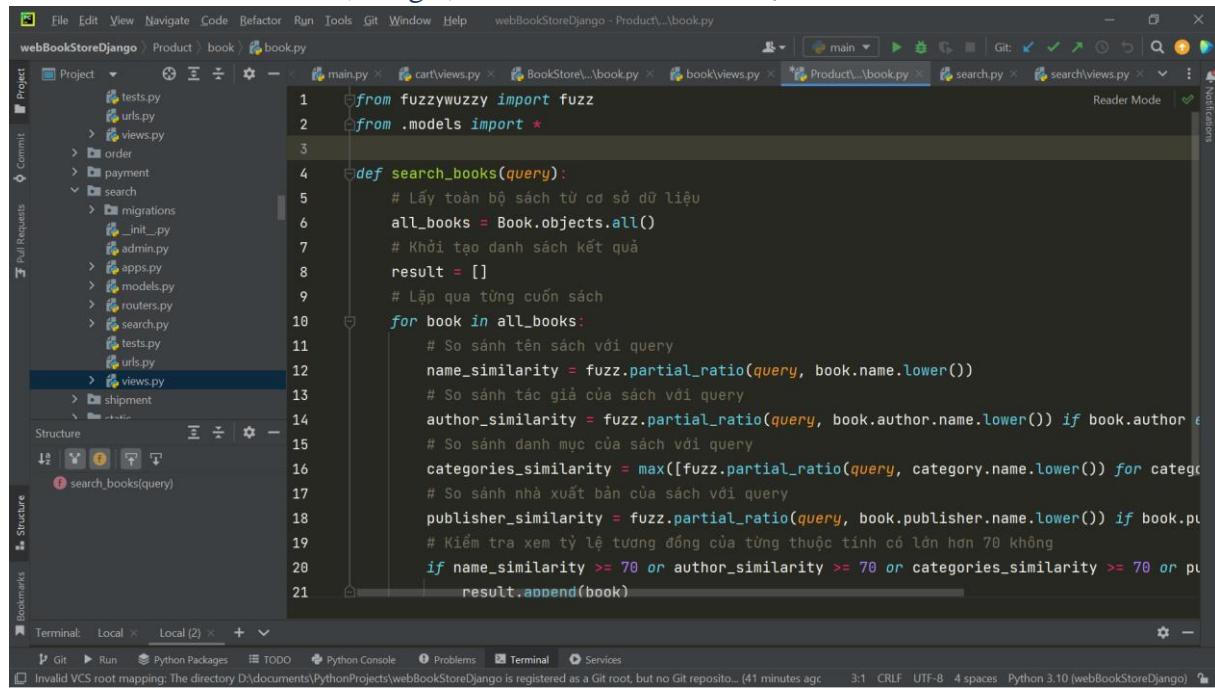


```

32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58

```

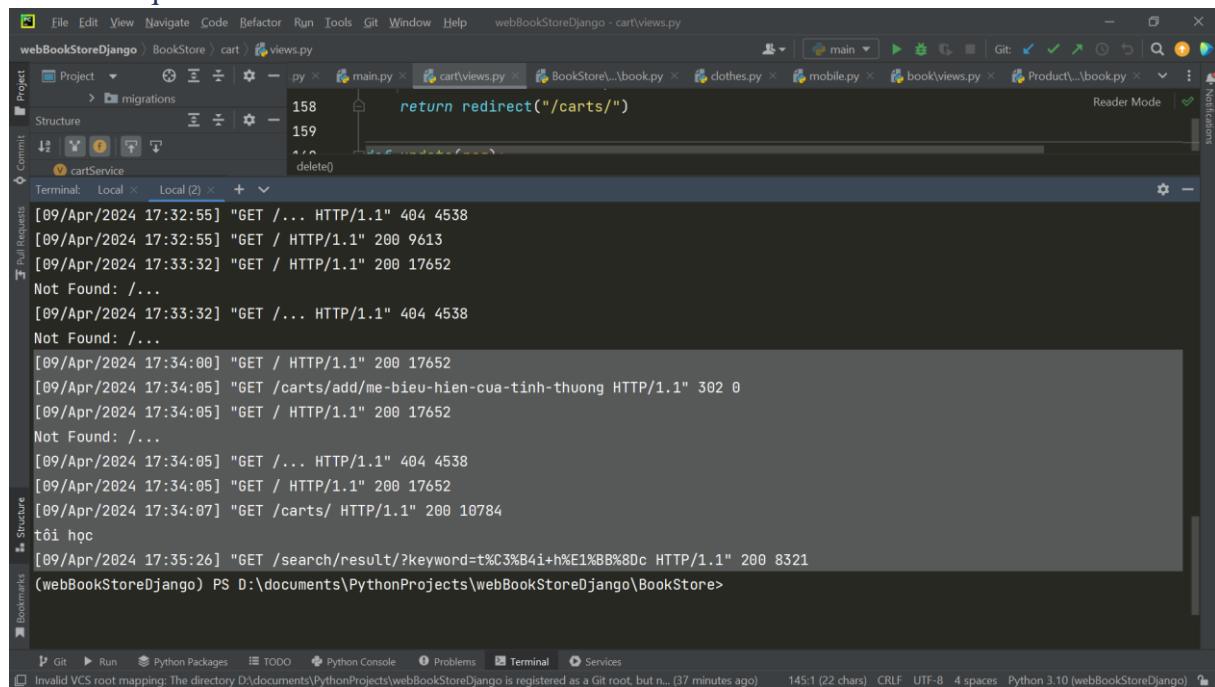
Tìm kiếm book theo name, tác giả, nhà xuất bản và danh mục



```
from fuzzywuzzy import fuzz
from .models import *

def search_books(query):
    # Lấy toàn bộ sách từ cơ sở dữ liệu
    all_books = Book.objects.all()
    # Khởi tạo danh sách kết quả
    result = []
    # Lặp qua từng cuốn sách
    for book in all_books:
        # So sánh tên sách với query
        name_similarity = fuzz.partial_ratio(query, book.name.lower())
        # So sánh tác giả của sách với query
        author_similarity = fuzz.partial_ratio(query, book.author.name.lower()) if book.author else 0
        # So sánh danh mục của sách với query
        categories_similarity = max([fuzz.partial_ratio(query, category.name.lower()) for category in book.categories.all()])
        # So sánh nhà xuất bản của sách với query
        publisher_similarity = fuzz.partial_ratio(query, book.publisher.name.lower()) if book.publisher else 0
        # Kiểm tra xem tỷ lệ tương đồng của từng thuộc tính có lớn hơn 70 không
        if name_similarity >= 70 or author_similarity >= 70 or categories_similarity >= 70 or publisher_similarity >= 70:
            result.append(book)
```

Demo kết quả

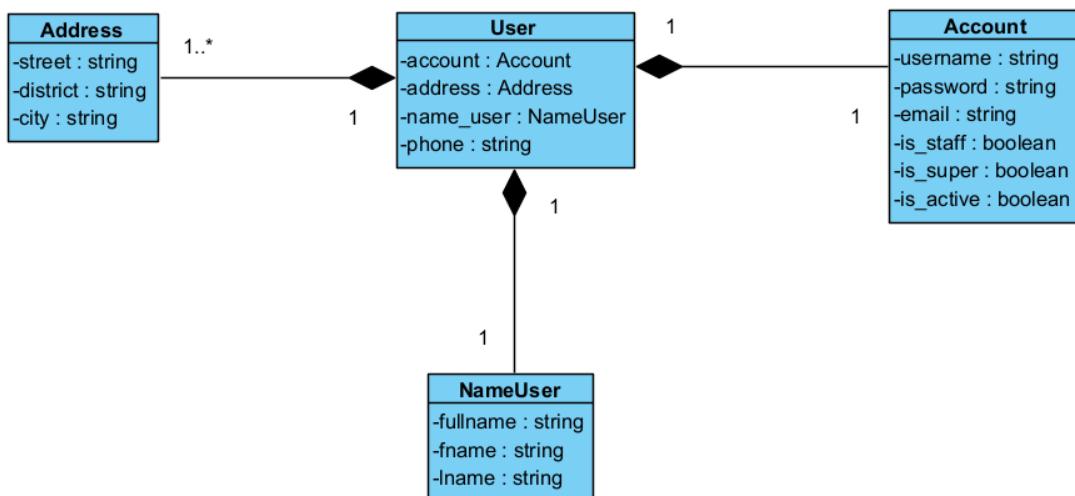


```
[09/Apr/2024 17:32:55] "GET /... HTTP/1.1" 404 4538
[09/Apr/2024 17:32:55] "GET / HTTP/1.1" 200 9613
[09/Apr/2024 17:33:32] "GET / HTTP/1.1" 200 17652
Not Found: /...
[09/Apr/2024 17:33:32] "GET /... HTTP/1.1" 404 4538
Not Found: /...
[09/Apr/2024 17:34:00] "GET / HTTP/1.1" 200 17652
[09/Apr/2024 17:34:05] "GET /carts/add/me-bieu-hien-cua-tinh-thuong HTTP/1.1" 302 0
[09/Apr/2024 17:34:05] "GET / HTTP/1.1" 200 17652
Not Found: /...
[09/Apr/2024 17:34:05] "GET /... HTTP/1.1" 404 4538
[09/Apr/2024 17:34:05] "GET / HTTP/1.1" 200 17652
[09/Apr/2024 17:34:07] "GET /carts/ HTTP/1.1" 200 10784
tôi học
[09/Apr/2024 17:35:26] "GET /search/result/?keyword=t%C3%B4i+h%C1%BB%80c HTTP/1.1" 200 8321
(webBookStoreDjango) PS D:\documents\PythonProjects\webBookStoreDjango\BookStore>
```

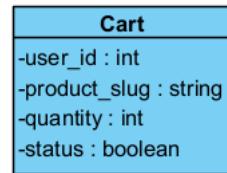
6. Xây dựng biểu đồ activity cho tương tác giữa các services

7. Xây dựng Biểu đồ lớp Phân tích cho từng service riêng lẻ

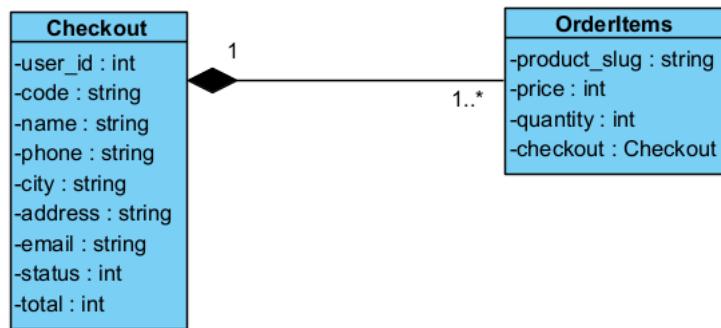
1. UserService



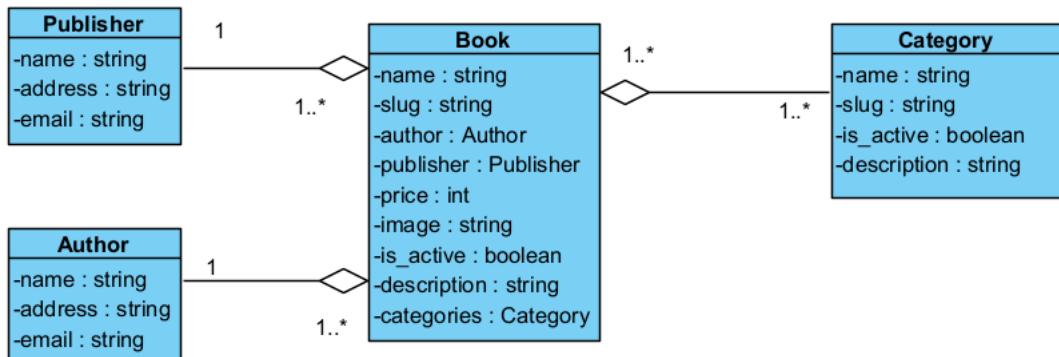
2. CartService



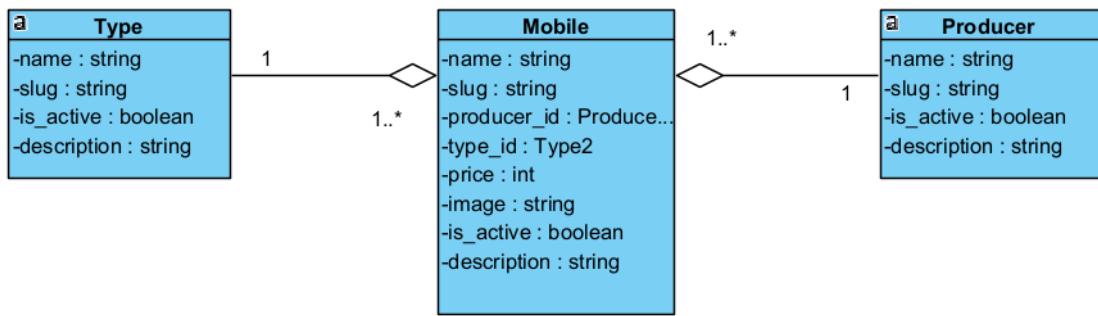
3. OrderService



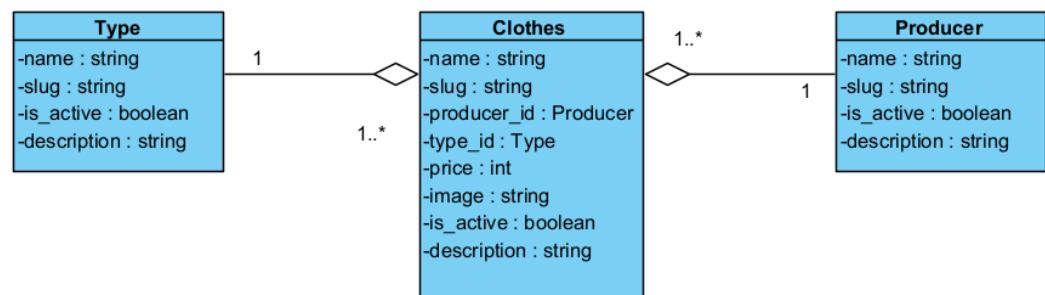
4. BookService



5. MobileService



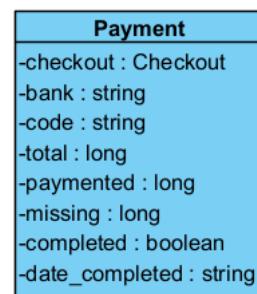
6. ClothesService



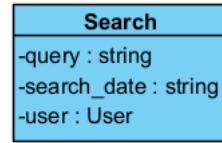
7. ShipmentService



8. PaymentService



9. SearchService



8. Xây dựng data model cho từng service và công nghệ phát triển tương ứng (sử dụng cả 3 mySQL, PostgreSQL, mongoDB)

1. CartService

```

class Cart(models.Model):
    cart_id = models.CharField(max_length=50, null=True)
    user_id = models.BigIntegerField(null=True, blank=True)
    product_slug = models.SlugField(null=True)
    quantity = models.IntegerField(default=1, blank=True)
    status = models.BooleanField(default=False)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
  
```

```

    "default": {
        "ENGINE": "django.db.backends.mysql",
        "NAME": "library",
        "USER": "root",
        # "PASSWORD": "anhgdt1710",
        "PASSWORD": "anh1710gdt",
        "HOST": "127.0.0.1",
        "PORT": "3306",
    },
  
```

```
class CartRouter:
    route_app_labels = {"cart"}
    db_name = "default"
    # truy vấn đọc (SELECT)
    def db_for_read(self, model, **hints):
        if model._meta.app_label in self.route_app_labels:
            return self.db_name
        return None
    # truy vấn ghi (INSERT, UPDATE, DELETE)
    def db_for_write(self, model, **hints):
        if model._meta.app_label in self.route_app_labels:
            return self.db_name
        return None
    # cho phép quan hệ (relationship) giữa hai đối tượng (obj1 và obj2)
    def allow_relation(self, obj1, obj2, **hints):...
    # thực hiện các migration trên database
    def allow_migrate(self, db, app_label, model_name=None, **hints):...
```

2. UserService

```
class Address(models.Model):  
    street = models.CharField(max_length=255, null=True)  
    district = models.CharField(max_length=255, null=True)  
    city = models.CharField(max_length=255, null=True)  
    note = models.TextField(default=None, null=True)  
    created_at = models.DateTimeField(auto_now_add=True)  
    updated_at = models.DateTimeField(auto_now=True)
```

```
    def __str__(self):...
```

```
class NameUser(models.Model):  
    fullname = models.CharField(max_length=255, null=True)  
    fname = models.CharField(max_length=255, null=True)  
    lname = models.CharField(max_length=255, null=True)  
    created_at = models.DateTimeField(auto_now_add=True)  
    updated_at = models.DateTimeField(auto_now=True)
```

```
    def __str__(self):...
```

```
class Account(models.Model):  
    username = models.CharField(max_length=255, unique=True)  
    password = models.CharField(max_length=255, null=True)  
    email = models.EmailField(max_length=255, null=True)  
    is_staff = models.BooleanField(default=False)  
    is_superuser = models.BooleanField(default=False)  
    is_active = models.BooleanField(default=True)  
    created_at = models.DateTimeField(auto_now_add=True)  
    updated_at = models.DateTimeField(auto_now=True)
```

```
    def __str__(self):...
```

```
class User(models.Model):  
    account = models.OneToOneField(Account, on_delete=models.SET_NULL, null=True)  
    address = models.OneToOneField(Address, on_delete=models.SET_NULL, null=True)  
    name_user = models.OneToOneField(NameUser, on_delete=models.SET_NULL, null=True)  
    phone = models.CharField(max_length=12, null=True)  
    created_at = models.DateTimeField(auto_now_add=True)  
    updated_at = models.DateTimeField(auto_now=True)
```

```

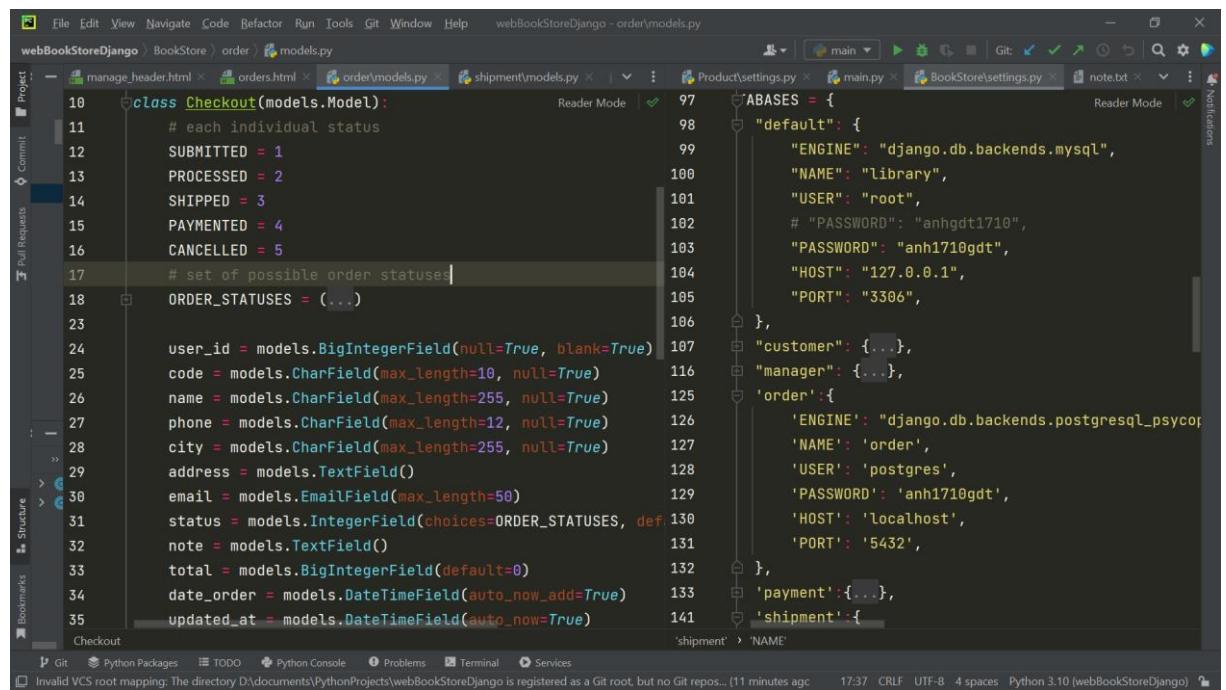
'default': {

    "ENGINE": "django.db.backends.mysql",
    "NAME": "user_service",
    "USER": "root",
    # "PASSWORD": "anhgdt1710",
    "PASSWORD": "anh1710gdt",
    "HOST": "127.0.0.1",
    "PORT": "3306",
}

```

3. OrderService

class Checkout



The screenshot shows the PyCharm IDE interface with two open files: `models.py` and `settings.py`.

models.py (Left Panel):

```

class Checkout(models.Model):
    # each individual status
    SUBMITTED = 1
    PROCESSED = 2
    SHIPPED = 3
    PAYMENTED = 4
    CANCELLED = 5

    # set of possible order statuses
    ORDER_STATUSES = (...)

    user_id = models.BigIntegerField(null=True, blank=True)
    code = models.CharField(max_length=10, null=True)
    name = models.CharField(max_length=255, null=True)
    phone = models.CharField(max_length=12, null=True)
    city = models.CharField(max_length=255, null=True)
    address = models.TextField()
    email = models.EmailField(max_length=50)
    status = models.IntegerField(choices=ORDER_STATUSES, default=0)
    note = models.TextField()
    total = models.BigIntegerField(default=0)
    date_order = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

```

settings.py (Right Panel):

```

DATABASES = {
    "default": {
        "ENGINE": "django.db.backends.mysql",
        "NAME": "library",
        "USER": "root",
        # "PASSWORD": "anhgdt1710",
        "PASSWORD": "anh1710gdt",
        "HOST": "127.0.0.1",
        "PORT": "3306",
    },
    "customer": {...},
    "manager": {...},
    "order": {
        "ENGINE": "django.db.backends.postgresql_psycopg2",
        "NAME": "order",
        "USER": "postgres",
        "PASSWORD": "anh1710gdt",
        "HOST": "localhost",
        "PORT": "5432",
    },
    "payment": {...},
    "shipment": {
        "NAME": ...
    }
}

```

class OrderItems

The screenshot shows the PyCharm IDE interface with several code files open in tabs at the top: `manage_header.html`, `orders.html`, `order\models.py`, `shipment\models.py`, `payment\models.py`, `search\`, `main.py`, and `BookStore\settings.py`. The `order\models.py` file is the active file, displaying Python code for a `OrderItems` model. The `BookStore\settings.py` file is also visible on the right, showing a `BASES` configuration block. The bottom status bar indicates an invalid VCS root mapping and the Python version (3.10).

```

class OrderRouter:
    route_app_labels = {"order"}
    db_name = "order"
    # truy vấn đọc (SELECT)
    def db_for_read(self, model, **hints):
        if model._meta.app_label in self.route_app_labels:
            return self.db_name
        return 'default'
    # truy vấn ghi (INSERT, UPDATE, DELETE)
    def db_for_write(self, model, **hints):
        if model._meta.app_label in self.route_app_labels:
            return self.db_name
        return 'default'

    # cho phép quan hệ (relationship) giữa hai đối tượng (obj1 và obj2)
    def allow_relation(self, obj1, obj2, **hints):
        ...

    # thực hiện các migration trên database
    def allow_migrate(self, db, app_label, model_name=None, **hints):
        ...

```

4. BookService

```
class Category(models.Model):  
    name = models.CharField(max_length=255)  
    slug = models.SlugField(max_length=255, unique=True,  
                           help_text='Unique value for product page URL,  
                           description = models.TextField(null=True)  
    is_active = models.BooleanField(default=True)  
    created_at = models.DateTimeField(auto_now_add=True)  
    updated_at = models.DateTimeField(auto_now=True)  
  
    def __str__(self):...  
  
class Author(models.Model):  
    name = models.CharField(max_length=255)  
    address = models.CharField(max_length=255, null=True)  
    email = models.EmailField(null=True)  
    created_at = models.DateTimeField(auto_now_add=True)  
    updated_at = models.DateTimeField(auto_now=True)
```

```
class Publisher(models.Model):  
    name = models.CharField(max_length=255)  
    address = models.CharField(max_length=255, null=True)  
    email = models.EmailField(null=True)  
    created_at = models.DateTimeField(auto_now_add=True)  
    updated_at = models.DateTimeField(auto_now=True)  
  
    def __str__(self):...
```

```

class Book(models.Model):
    name = models.CharField(max_length=255, unique=True)
    slug = models.SlugField(max_length=255, unique=True,
                           help_text='Unique value for product page URL, created from name.', null=True)
    author = models.ForeignKey(Author, on_delete=models.SET_NULL, null=True)
    publisher = models.ForeignKey(Publisher, on_delete=models.SET_NULL, null=True)
    price = models.IntegerField(default=0)
    old_price = models.IntegerField(default=0)
    image = models.TextField()
    # thumbnail = models.ImageField(upload_to='images/products-thumbnails')
    is_active = models.BooleanField(default=True)
    is_bestseller = models.BooleanField(default=False)
    description = models.TextField(null=True)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    # categories = models.ArrayReferenceField(to=Category, on_delete=models.CASCADE,
    categories = models.ManyToManyField(to=Category)

```

```

class BookRouter:
    route_app_labels = {"book"}
    db_name = "book"
    # truy vấn đọc (SELECT)
    def db_for_read(self, model, **hints):...
    # truy vấn ghi (INSERT, UPDATE, DELETE)
    def db_for_write(self, model, **hints):...

    # cho phép quan hệ (relationship) giữa hai đối tượng (obj1 và obj2)
    def allow_relation(self, obj1, obj2, **hints):...

    # thực hiện các migration trên database
    def allow_migrate(self, db, app_label, model_name=None, **hints):...

```

```

'book': {
    'ENGINE': 'django',
    'NAME': 'library',
    'HOST': 'localhost',
    'PORT': 27017,
},

```

5. MobileService

```
class Producer(models.Model):  
    name = models.CharField(max_length=255, null=True)  
    slug = models.SlugField(max_length=255, unique=True,  
                           help_text='Unique value for product page URL')  
    description = models.TextField(null=True)  
    is_active = models.BooleanField(default=True)  
    created_at = models.DateTimeField(auto_now_add=True)  
    updated_at = models.DateTimeField(auto_now=True)  
  
    def __str__(self):...  
  
class Type(models.Model):  
    name = models.CharField(max_length=255, null=True)  
    slug = models.SlugField(max_length=255, unique=True,  
                           help_text='Unique value for product page URL')  
    description = models.TextField(null=True)  
    is_active = models.BooleanField(default=True)  
    created_at = models.DateTimeField(auto_now_add=True)  
    updated_at = models.DateTimeField(auto_now=True)  
  
    def __str__(self):...  
  
class Phone(models.Model):  
    name = models.CharField(max_length=255, unique=True)  
    slug = models.SlugField(max_length=255, unique=True,  
                           help_text='Unique value for product page URL, created from  
producer_id = models.ForeignKey(Producer, models.CASCADE)  
type_id = models.ForeignKey(Type, models.CASCADE)  
# producer_id = models.EmbeddedField(model_container=Producer,  
# type_id = models.EmbeddedField(model_container=Type,)  
    price = models.IntegerField(default=0)  
    old_price = models.IntegerField(default=0)  
    # image = models.ImageField()  
    image = models.TextField(null=True)  
    is_active = models.BooleanField(default=True)  
    is_bestseller = models.BooleanField(default=False)  
    description = models.TextField(null=True)  
    created_at = models.DateTimeField(auto_now_add=True)  
    updated_at = models.DateTimeField(auto_now=True)
```

```
class MobileRouter:
    route_app_labels = {"mobile"}
    db_name = "mobile"
    # truy vấn đọc (SELECT)
    def db_for_read(self, model, **hints):
        # truy vấn ghi (INSERT, UPDATE, DELETE)
        def db_for_write(self, model, **hints):
            # cho phép quan hệ (relationship) giữa hai đối tượng (obj1 và obj2)
            def allow_relation(self, obj1, obj2, **hints):
                # thực hiện các migration trên database
                def allow_migrate(self, db, app_label, model_name=None, **hints):
                    'mobile': {
                        'ENGINE': 'djongo',
                        'NAME': 'mobile',
                        'HOST': 'localhost',
                        'PORT': 27017,
                    },
```

6. ClothesService

```
class Producer(models.Model):  
    name = models.CharField(max_length=255, null=True)  
    slug = models.SlugField(max_length=255, unique=True,  
                           help_text='Unique value for product page URL')  
    description = models.TextField(null=True)  
    is_active = models.BooleanField(default=True)  
    created_at = models.DateTimeField(auto_now_add=True)  
    updated_at = models.DateTimeField(auto_now=True)  
  
    def __str__(self):...  
  
class Type(models.Model):  
    name = models.CharField(max_length=255, null=True)  
    slug = models.SlugField(max_length=255, unique=True,  
                           help_text='Unique value for product page URL')  
    description = models.TextField(null=True)  
    is_active = models.BooleanField(default=True)  
    created_at = models.DateTimeField(auto_now_add=True)  
    updated_at = models.DateTimeField(auto_now=True)  
  
    def __str__(self):...
```

```
class ClothesRouter:  
    def db_for_read(self, model, **hints):  
        if model._meta.app_label == 'clothes':  
            # return 'mongodb'  
            return 'clothes'  
        return None  
  
    def db_for_write(self, model, **hints):  
        if model._meta.app_label == 'clothes':  
            # return 'mongodb'  
            return 'clothes'  
        return None  
  
    def allow_relation(self, obj1, obj2, **hints):...  
  
    def allow_migrate(self, db, app_label, model_name=None, **hints):...
```

```
'clothes': {  
    'ENGINE': 'djongo',  
    'NAME': 'clothes',  
    'HOST': 'localhost',  
    'PORT': 27017,  
},
```

7. ShipmentService

```
class Shipment(models.Model):  
    checkout = models.BigIntegerField(null=True, unique=True)  
    code = models.CharField(max_length=15, null=True)  
    shipper = models.CharField(max_length=255)  
    delivered = models.BooleanField(default=False)  
    date_shipment = models.DateTimeField(auto_now_add=True)  
    note = models.TextField()  
    create_at = models.DateTimeField(auto_now_add=True, null=True)  
    updated_at = models.DateTimeField(auto_now=True)
```

```
class ShipmentRouter:  
    route_app_labels = {"shipment"}  
    db_name = "shipment"  
  
    # truy vấn đọc (SELECT)  
    def db_for_read(self, model, **hints):...  
    # truy vấn ghi (INSERT, UPDATE, DELETE)  
    def db_for_write(self, model, **hints):...  
  
    # cho phép quan hệ (relationship) giữa hai đối tượng (obj1 và obj2)  
    def allow_relation(self, obj1, obj2, **hints):...  
  
    # thực hiện các migration trên database  
    def allow_migrate(self, db, app_label, model_name=None, **hints):...
```

```

'shipment':{
    'ENGINE': "django.db.backends.postgresql_psycopg2",
    'NAME': 'shipment',
    'USER': 'postgres',
    'PASSWORD': 'anh1710gdt',
    'HOST': 'localhost',
    'PORT': '5432',
},

```

8. PaymentService

```

class Payment(models.Model):
    bank = models.CharField(max_length=30, null=True)
    checkout = models.ForeignKey(null=True, blank=True) # one-to-one
    code = models.CharField(max_length=25, null=True)
    total = models.BigIntegerField(null=True)
    paymented = models.BigIntegerField(null=True)
    missing = models.BigIntegerField(null=True)
    completed = models.BooleanField(default=False)
    note = models.TextField()
    date_completed = models.DateTimeField(null=True)
    create_at = models.DateTimeField(auto_now_add=True, null=True)
    updated_at = models.DateTimeField(auto_now=True)

```

```

class PaymentRouter:
    route_app_labels = {"payment"}
    db_name = "payment"

    # truy vấn đọc (SELECT)
    def db_for_read(self, model, **hints):...

    # truy vấn ghi (INSERT, UPDATE, DELETE)
    def db_for_write(self, model, **hints):...

    # cho phép quan hệ (relationship) giữa hai đối tượng (obj1 và obj2)
    def allow_relation(self, obj1, obj2, **hints):...

    # thực hiện các migration trên database
    def allow_migrate(self, db, app_label, model_name=None, **hints):...

```

```
'payment': {
    'ENGINE': "django.db.backends.postgresql_psycopg2",
    'NAME': 'payment',
    'USER': 'postgres',
    'PASSWORD': 'anh1710gdt',
    'HOST': 'localhost',
    'PORT': '5432',
},
```

9. SearchService

```
class SearchTerm(models.Model):
    q = models.CharField(max_length=50)
    search_date = models.DateTimeField(auto_now_add=True)
    user = models.ForeignKey(User, on_delete=models.SET_NULL, null=True)

    def __str__(self) -> str:...
```



```
class SearchRouter:
    route_app_labels = {"search"}
    db_name = "search"
    # truy vấn đọc (SELECT)
    def db_for_read(self, model, **hints):...
    # truy vấn ghi (INSERT, UPDATE, DELETE)
    def db_for_write(self, model, **hints):...

    # cho phép quan hệ (relationship) giữa hai đối tượng (obj1 và obj2)
    def allow_relation(self, obj1, obj2, **hints):...
    # thực hiện các migration trên database
    def allow_migrate(self, db, app_label, model_name=None, **hints):...
```



```
'search': {
    'ENGINE': 'djongo',
    'NAME': 'search',
    'HOST': 'localhost',
    'PORT': 27017,
},
```

9. Xây dựng biểu đồ lớp thiết kế và kiến trúc cho từng service với MVT Django

10. Code và demo

1. CartService

Views.py

```
class cartService:
    def get_cart_service(url='http://127.0.0.1:9995/carts/api/cart/', user_id=0, product_slug=''):
    def update_cart_service(url='http://127.0.0.1:9995/carts/api/cart/', user_id=0, product_slug='', data={})
    def delete_cart_service(url='http://127.0.0.1:9995/carts/api/', user_id=0, product_slug='')
    def get_all_cart_service(url='http://127.0.0.1:9995/carts/api/', user_id=0)
    def create_cart_service(url='http://127.0.0.1:9995/carts/api/', user_id=0, data={})
    def allCart(req)
    def add(req, slug)
    def addMobile(req, slug)
    def addClothes(req, slug)
    def delete(req, slug)
    def update(req)
```

Urls.py

```
urlpatterns = [
    path('', views.allCart, name="cart"),
    path('add/<slug:slug>', views.add, name="add-cart"),
    path('add_mobile/<slug:slug>', views.addMobile, name="add-mobile-to-cart"),
    path('add_clothes/<slug:slug>', views.addClothes, name="add-clothes-to-cart"),
    path('delete/<slug:slug>', views.delete, name="delete-cart"),
    path('update', views.update, name='update-cart'),
    # api cart
    path('api/cart/<int:user_id>/<slug:product_slug>', service.get_and_update_cart),
    path('api/<int:user_id>/<slug:product_slug>/delete', service.delete_cart),
    path('api/<int:user_id>/create', service.get_carts_and_create_cart),
]
```

Service.py

```
# API để lấy giỏ hàng theo user_id và product_slug và status=False
@csrf_exempt
def get_and_update_cart(request, user_id, product_slug): ...

# API để xóa giỏ hàng
@csrf_exempt
def delete_cart(request, user_id, product_slug): ...

# API để tạo giỏ hàng mới
@csrf_exempt
def get_carts_and_create_cart(request, user_id): ...
```

2. UserService

Views.py:

Lấy thông tin, cập nhập và thay đổi mật khẩu người dùng

```
@csrf_exempt
def informations(request, id):
    method = request.method
    if method == "GET": ...
    if method == "POST": ...

# View để thay đổi mật khẩu (POST: thay đổi mật khẩu)
@csrf_exempt
def change_password(request, id):
    method = request.method
    if method == "POST": ...
```

Đăng ký và đăng nhập người dùng

```
# View để đăng ký người dùng
@csrf_exempt
def register(request):
    if request.method == "POST": ...

# View để đăng nhập
@csrf_exempt
def login(request):
    if request.method == "POST": ...
```

Urls.py

```
from . import views
from django.urls import path
|
urlpatterns = [
    path('user/informations/<int:id>', views.informations, name="informations"),
    path('user/change-password/<int:id>', views.change_password, name="change-password")
]

urlpatterns = [
    path('user/register', views.register, name="register"),
    path('user/login', views.login, name="login"),
]
```

3. OrderService

Views.py

```

get_checkouts_by_user_service(url='http://127.0.0.1:9995/orders/api/checkouts/', user_id=0)
get_checkouts_service(url='http://127.0.0.1:9995/orders/api/checkouts')
get_order_items_service(url='http://127.0.0.1:9995/orders/api/order-items/', checkout_id=0)
get_all_order_items_service(url='http://127.0.0.1:9995/orders/api/all-order-items')
create_order_service(url='http://127.0.0.1:9995/orders/api/order/create', data={})
cancel_checkout_service(url='http://127.0.0.1:9995/orders/api/checkouts/', id=0)
re_order_checkout_service(url='http://127.0.0.1:9995/orders/api/checkouts/', id=0)
get_checkout_service(url='http://127.0.0.1:9995/orders/api/checkout/', checkout_id=0)
getOrders(request)
checkout(request)
detailsOrder(request, id)
cancelOrder(request, id)
re_Order(request, id)

```

Urls.py

```

urlpatterns = [
    path('', getOrders, name="orders"),
    path('checkout/', checkout, name="checkout"),
    path('details/<int:id>', detailsOrder, name="details-order"),
    path('cancel-order/<int:id>', cancelOrder, name="cancel-order"),
    path('re-order/<int:id>', re_Order, name="re-order"),
]

# api order
path('api/order/create', service.create_order),
path('api/checkouts/<int:user_id>', service.get_checkouts_by_user),
path('api/checkouts', service.get_checkouts),
path('api/checkouts/<int:id>/cancel', service.cancel_checkout),
path('api/checkouts/<int:id>/re_order', service.re_order_checkout),
path('api/order-items/<int:checkout_id>', service.get_order_items),
path('api/checkout/<int:checkout_id>', service.get_checkout),
path('api/all-order-items', service.get_all_order_items),
]

```

Service.py

```

f _generate_code()
f validate_data(data)
f create_order(request)
f get_checkouts_by_user(request, user_id)
f get_checkouts(request)
f cancel_checkout(request, id)
f re_order_checkout(request, id)
f get_order_items(request, checkout_id)
f get_all_order_items(request)
f get_checkout(request, checkout_id)

```

4. BookService

Views.py

```
❷ BookListCreateView(generics.ListCreateAPIView)
❷ BookRetrieveUpdateDestroyAPIViewID(generics.RetrieveUpdateDestroyAPIView)
❷ BookRetrieveUpdateDestroyAPIViewSLUG(generics.RetrieveUpdateDestroyAPIView)
❷ CategoryListCreateView(generics.ListCreateAPIView)
❷ CategoryRetrieveUpdateDestroyAPIViewID(generics.RetrieveUpdateDestroyAPIView)
❷ CategoryRetrieveUpdateDestroyAPIViewSLUG(generics.RetrieveUpdateDestroyAPIView)
❷ ShowAllBookByCategory(APIView)
❷ BookSearchAPIView(APIView)
```

Urls.py

```
urlpatterns = [
    path('', BookListCreateView.as_view(), name='book-list-create'),
    path('<int:id>', BookRetrieveUpdateDestroyAPIViewID.as_view(), name='book-detail-id'),
    path('slug/<slug:slug>', BookRetrieveUpdateDestroyAPIViewSLUG.as_view(), name='book-d'),
    path('categories/', CategoryListCreateView.as_view(), name='book-category-list-create'),
    path('categories/<int:id>', CategoryRetrieveUpdateDestroyAPIViewID.as_view(), name='b'),
    path('categories/slug/<slug:slug>', CategoryRetrieveUpdateDestroyAPIViewSLUG.as_view()),
    path('books-by-category/<int:id>', ShowAllBookByCategory.as_view(), name='show-books-'),
    path('search/', BookSearchAPIView.as_view(), name='book-search'),
]
```

5. MobileService

Views.py

```
# get list, post
class PhoneListCreateView(generics.ListCreateAPIView): ...

# get, put, delete by id
class PhoneRetrieveUpdateDestroyAPIViewID(generics.RetrieveUpdateDestroyAPIView): ...

# get, put, delete by slug
class PhoneRetrieveUpdateDestroyAPIViewSLUG(generics.RetrieveUpdateDestroyAPIView): ...


class ProducerListCreateView(generics.ListCreateAPIView): ...

# get, put, delete by id
class ProducerRetrieveUpdateDestroyAPIViewID(generics.RetrieveUpdateDestroyAPIView): ...

# get, put, delete by slug
class ProducerRetrieveUpdateDestroyAPIViewSLUG(generics.RetrieveUpdateDestroyAPIView)

class TypeListCreateView(generics.ListCreateAPIView): ...
```

```

# get, put, delete by id
class TypeRetrieveUpdateDestroyAPIViewID(generics.RetrieveUpdateDestroyAPIView): ...

# get, put, delete by slug
class TypeRetrieveUpdateDestroyAPIViewSLUG(generics.RetrieveUpdateDestroyAPIView): ...

class ShowAllMobilesByType(APIView):
    def get(self, request, id): ...

class MobileSearchAPIView(APIView):
    def get(self, request): ...

```

Urls.py

```

urlpatterns = [
    path('', PhoneListCreateView.as_view(), name='phone-list-create'),
    path('<int:id>', PhoneRetrieveUpdateDestroyAPIViewID.as_view(), name='phone-detail'),
    path('slug/<slug:slug>', PhoneRetrieveUpdateDestroyAPIViewSLUG.as_view(), name='phone'),
    path('producers/', ProducerListCreateView.as_view(), name='mobile-producer-list-create'),
    path('producers/<int:id>', ProducerRetrieveUpdateDestroyAPIViewID.as_view(), name='mobile-producer'),
    path('producers.slug/<slug:slug>', ProducerRetrieveUpdateDestroyAPIViewSLUG.as_view(), name='mobile-producer'),
    path('types/', TypeListCreateView.as_view(), name='mobile-type-list-create'),
    path('types/<int:id>', TypeRetrieveUpdateDestroyAPIViewID.as_view(), name='mobile-type'),
    path('types.slug/<slug:slug>', TypeRetrieveUpdateDestroyAPIViewSLUG.as_view(), name='mobile-type'),
    path('mobiles-by-type/<int:id>', ShowAllMobilesByType.as_view(), name='show-mobiles'),
    path('search/', MobileSearchAPIView.as_view(), name='mobile-search'),
]

```

6. ClothesService

Views.py

- ⌚ ClothesListCreateView(generics.ListCreateAPIView)
- ⌚ ClothesRetrieveUpdateDestroyAPIViewID(generics.RetrieveUpdateDestroyAPIView)
- ⌚ ClothesRetrieveUpdateDestroyAPIViewSLUG(generics.RetrieveUpdateDestroyAPIView)
- ⌚ ProducerListCreateView(generics.ListCreateAPIView)
- ⌚ ProducerRetrieveUpdateDestroyAPIViewID(generics.RetrieveUpdateDestroyAPIView)
- ⌚ ProducerRetrieveUpdateDestroyAPIViewSLUG(generics.RetrieveUpdateDestroyAPIView)
- ⌚ TypeListCreateView(generics.ListCreateAPIView)
- ⌚ TypeRetrieveUpdateDestroyAPIViewID(generics.RetrieveUpdateDestroyAPIView)
- ⌚ TypeRetrieveUpdateDestroyAPIViewSLUG(generics.RetrieveUpdateDestroyAPIView)
- ⌚ ShowAllClothesByType(APIView)
- ⌚ ClothesSearchAPIView(APIView)

Urls.py

```
urlpatterns = [
    path('', ClothesListCreateView.as_view(), name='clothes-list-create'),
    path('<int:id>/', ClothesRetrieveUpdateDestroyAPIViewID.as_view(), name='clothes-det'),
    path('slug/<slug:slug>/', ClothesRetrieveUpdateDestroyAPIViewSLUG.as_view(), name='c'),
    path('producers/', ProducerListCreateView.as_view(), name='clothes-producer-list-cre'),
    path('producers/<int:id>/', ProducerRetrieveUpdateDestroyAPIViewID.as_view(), name='p'),
    path('producers/slug/<slug:slug>/', ProducerRetrieveUpdateDestroyAPIViewSLUG.as_view()),
    path('types/', TypeListCreateView.as_view(), name='clothes-type-list-create'),
    path('types/<int:id>/', TypeRetrieveUpdateDestroyAPIViewID.as_view(), name='clothes-'),
    path('types/slug/<slug:slug>/', TypeRetrieveUpdateDestroyAPIViewSLUG.as_view(), name='t'),
    path('clothes-by-type/<int:id>/', ShowAllClothesByType.as_view(), name='show-clothes'),
    path('search/', ClothesSearchAPIView.as_view(), name='clothes-search'),
```

7. ShipmentService

8. PaymentService

views.py

```
    f create_order_service(url='http://127.0.0.1:9995/orders/api/order/create', data={})
    f create_or_update_payment_service(url='http://127.0.0.1:9995/payment/api/create_or_update', data={})
    f get_payment_by_checkoutid_service(url='http://127.0.0.1:9995/payment/api/get/', checkout_id=0)
    f paymentBank(request)
```

Urls.py

```
urlpatterns = [
    path('', paymentBank, name="payment-bank"),

    # api payment
    path('api/create_or_update', service.create_or_update_payment),
    path('api/get/<int:checkout_id>', service.get_payment_by_checkoutid),

]
```

Service.py

```

def validate_bank(bank):...

# API để tạo payment mới
@csrf_exempt
def create_or_update_payment(request):...

# API để lấy thông tin của 1 payment by checkout_id
@csrf_exempt
def get_payment_by_checkoutid(request, checkout_id):...

```

9. SearchService

views.py

```

def search(req):
    query = req.GET.get('keyword')
    products = {}
    if query:...
    else:...
    context = {
        'page_title': 'result search for product',
        'products': products['products'],
        'mobiles': products['mobiles'],
        'clothes': products['clothes'],
        'query': query,
    }

    return render(req, 'result_search.html', context)

```

Urls.py

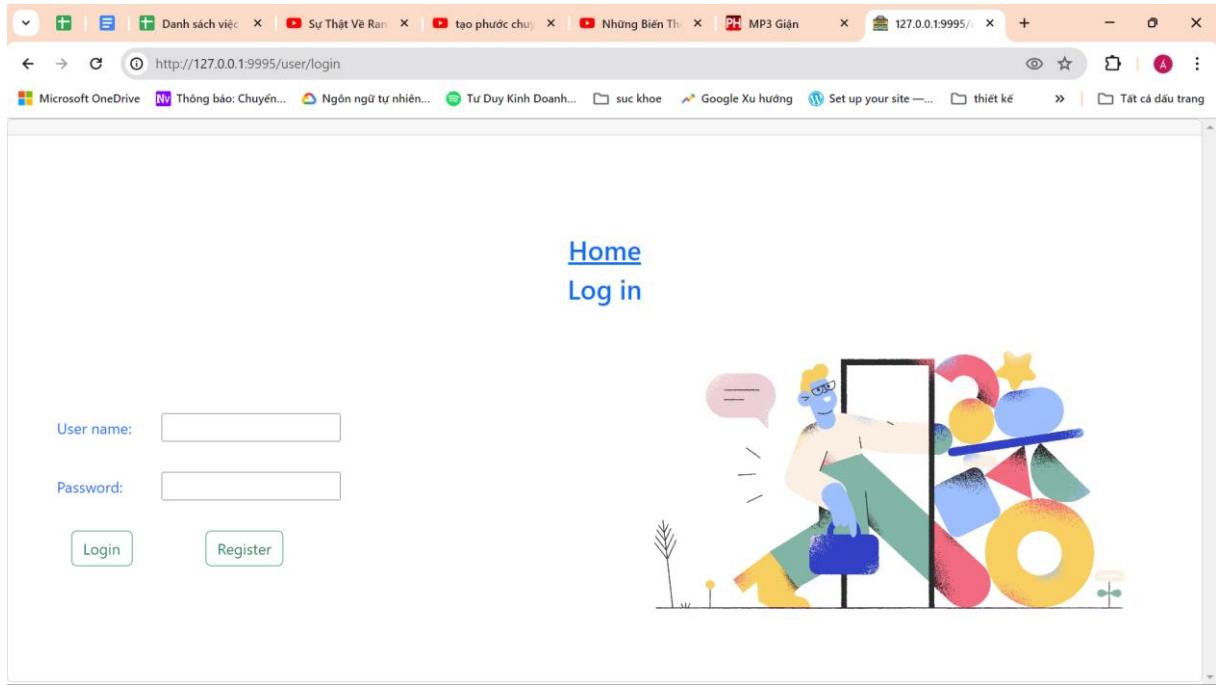
```

urlpatterns = [
    # path('', views.home, name="home"),
    path('search/result/', views.search, name="search"),
    path('search_book/result/', views.search, name="search-book"),
    path('search_mobile/result/', views.search, name="search-mobile"),
    path('search_clothes/result/', views.search, name="search-clothes"),
]

```

10. Demo

Login user:



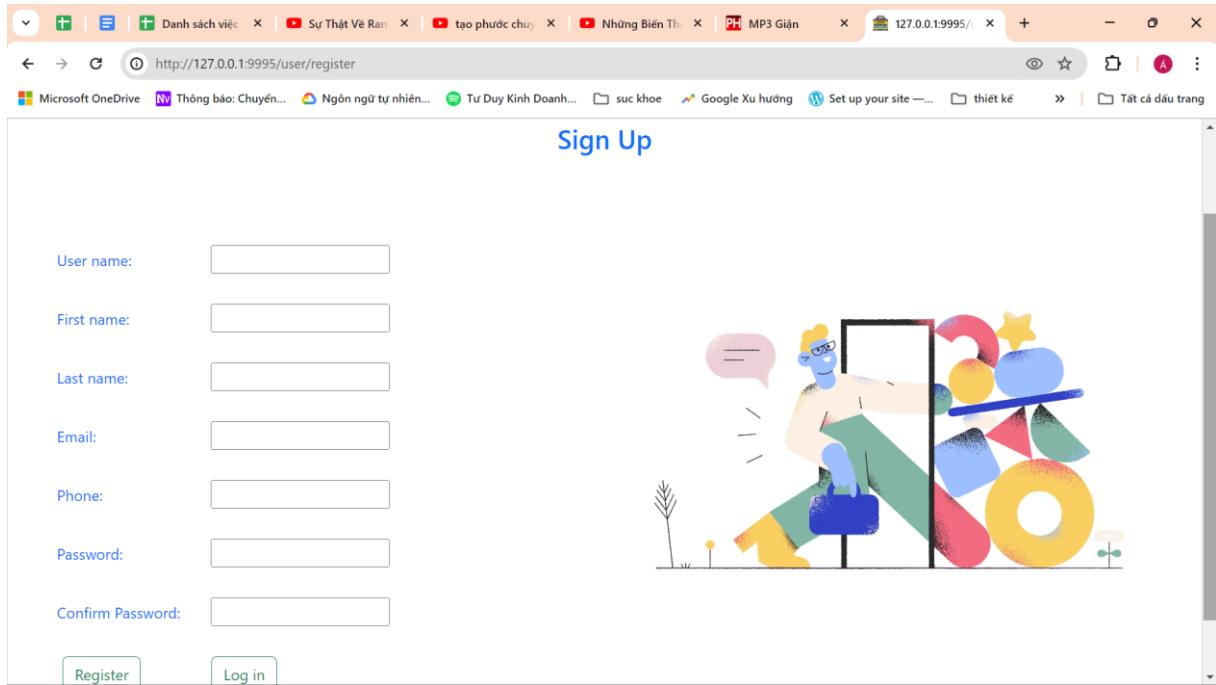
Home
Log in

User name:

Password:

[Login](#) [Register](#)

Signup user



Sign Up

User name:

First name:

Last name:

Email:

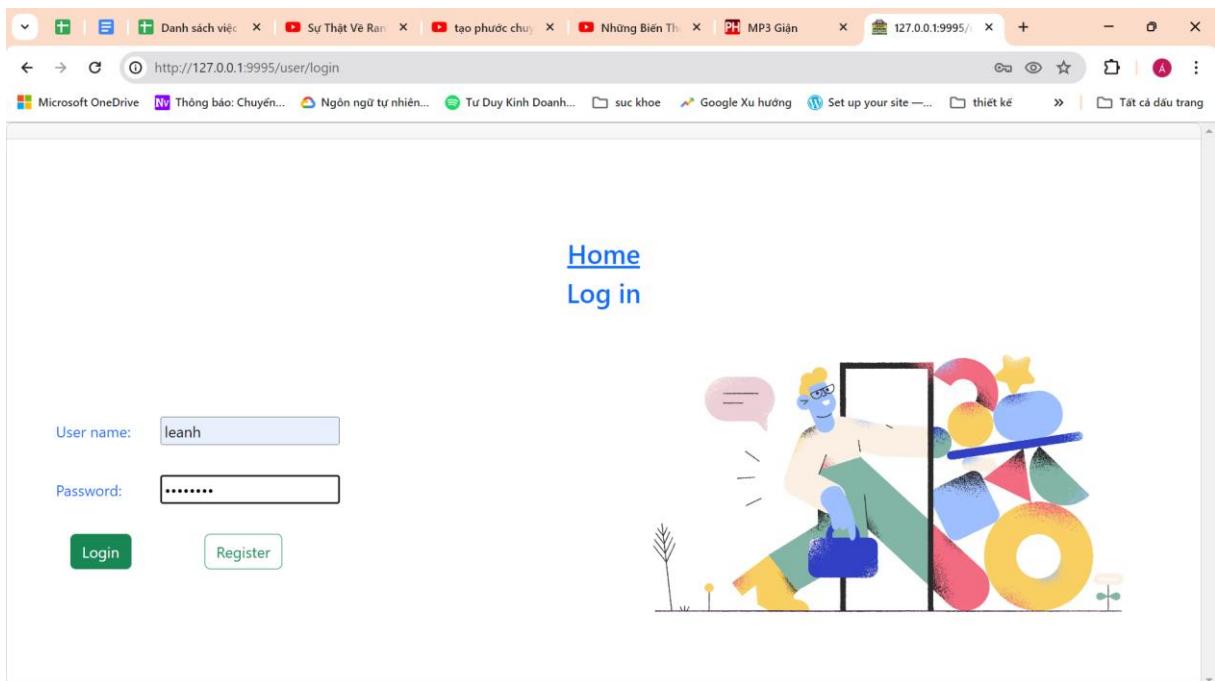
Phone:

Password:

Confirm Password:

[Register](#) [Log in](#)

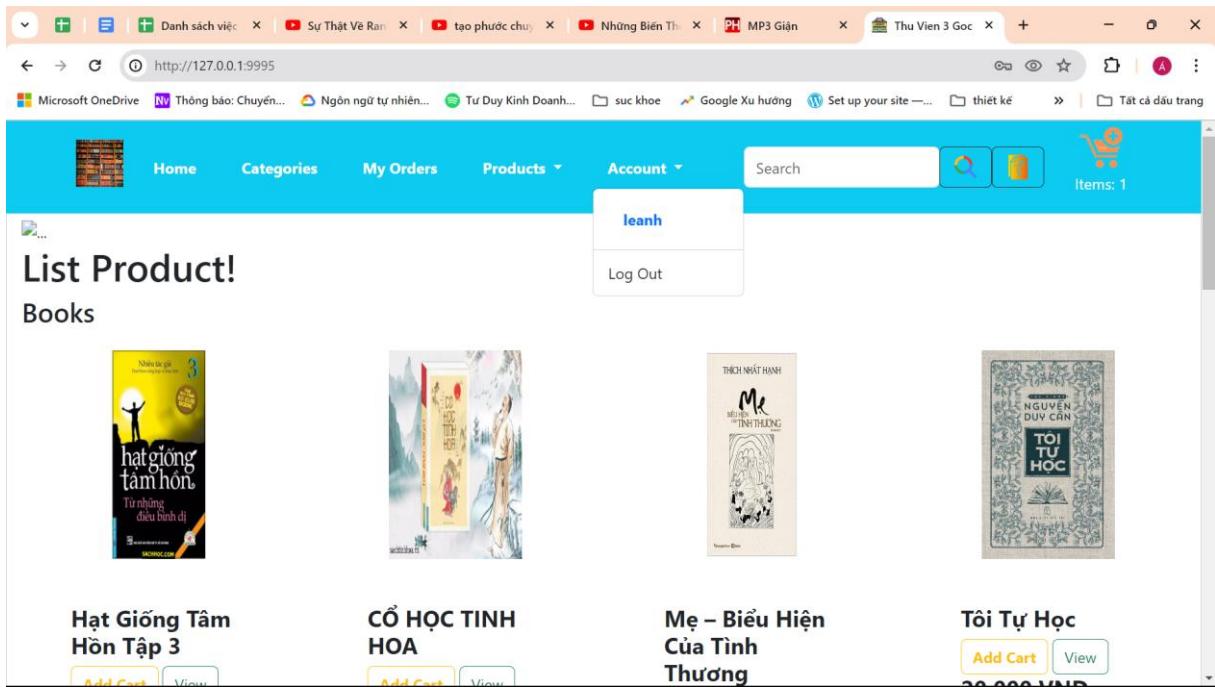
User login:



[23/Apr/2024 20:41:42] "POST /user/login HTTP/1.1" 200 879

[23/Apr/2024 20:41:42] "POST /user/login HTTP/1.1" 302 0

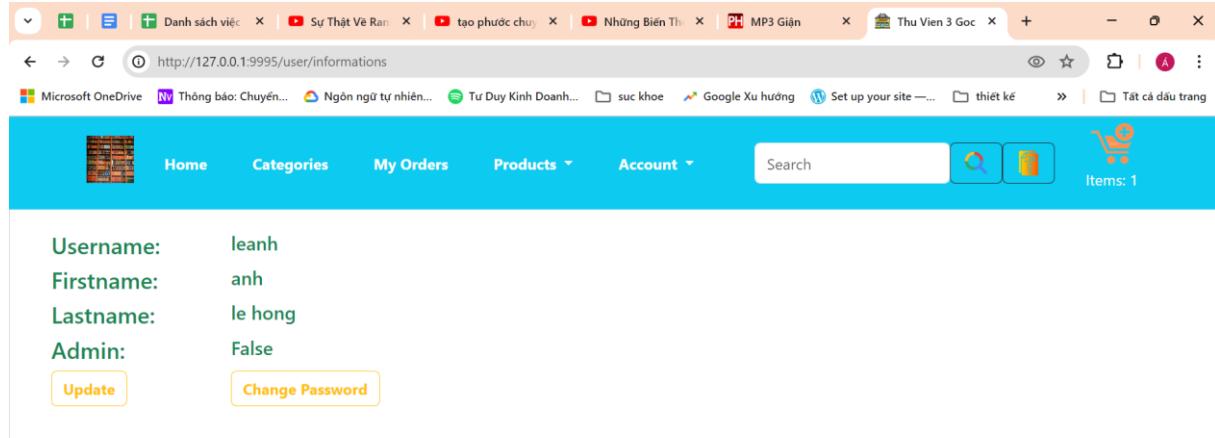
Home Customer



Gọi api product

```
[23/Apr/2024 20:41:42] "GET /books/ HTTP/1.1" 200 1638
[23/Apr/2024 20:41:42] "GET /mobiles/ HTTP/1.1" 200 1989
[23/Apr/2024 20:41:42] "GET /clothes/ HTTP/1.1" 200 859
```

Xem thông tin user

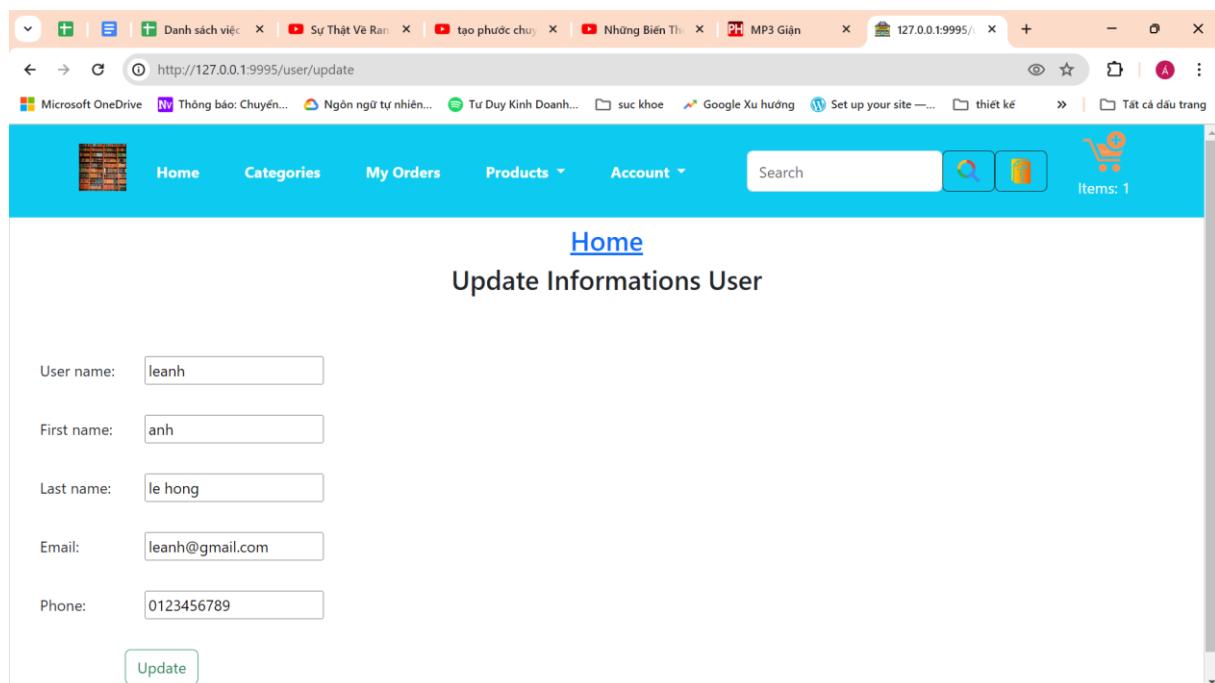


The screenshot shows a web browser window with the URL <http://127.0.0.1:9995/user/informations>. The page displays user details for a user named 'leanh'. The details are as follows:

| | |
|------------|---------|
| Username: | leanh |
| Firstname: | anh |
| Lastname: | le hong |
| Admin: | False |

Below the details are two buttons: 'Update' and 'Change Password'.

Update user

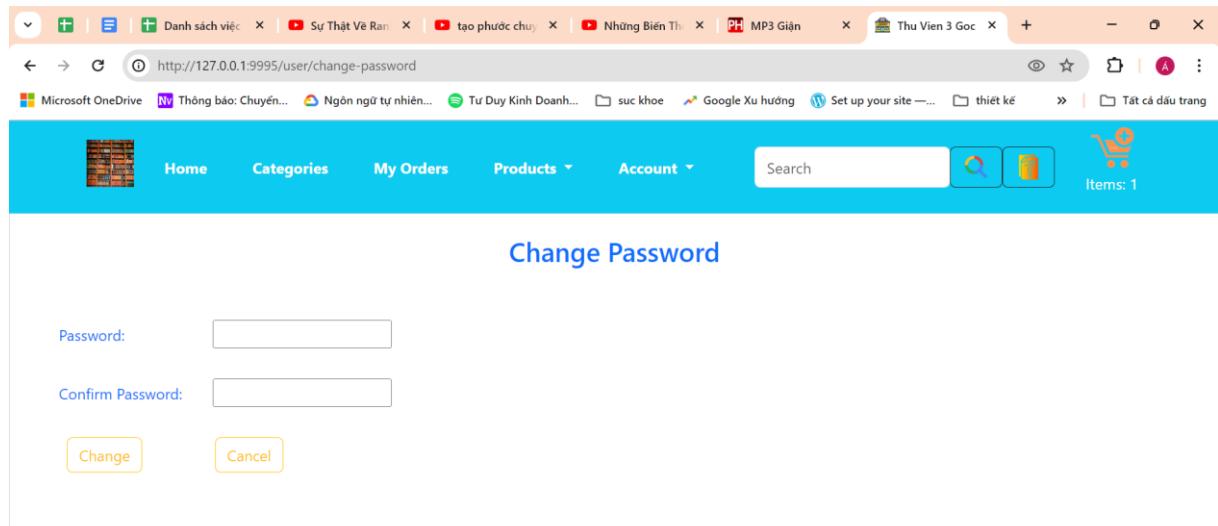


The screenshot shows a web browser window with the URL <http://127.0.0.1:9995/user/update>. The page title is 'Home' and the sub-section is 'Update Informations User'. The form fields are:

| | |
|-------------|-----------------|
| User name: | leanh |
| First name: | anh |
| Last name: | le hong |
| Email: | leanh@gmail.com |
| Phone: | 0123456789 |

Below the form is a 'Update' button.

Change password

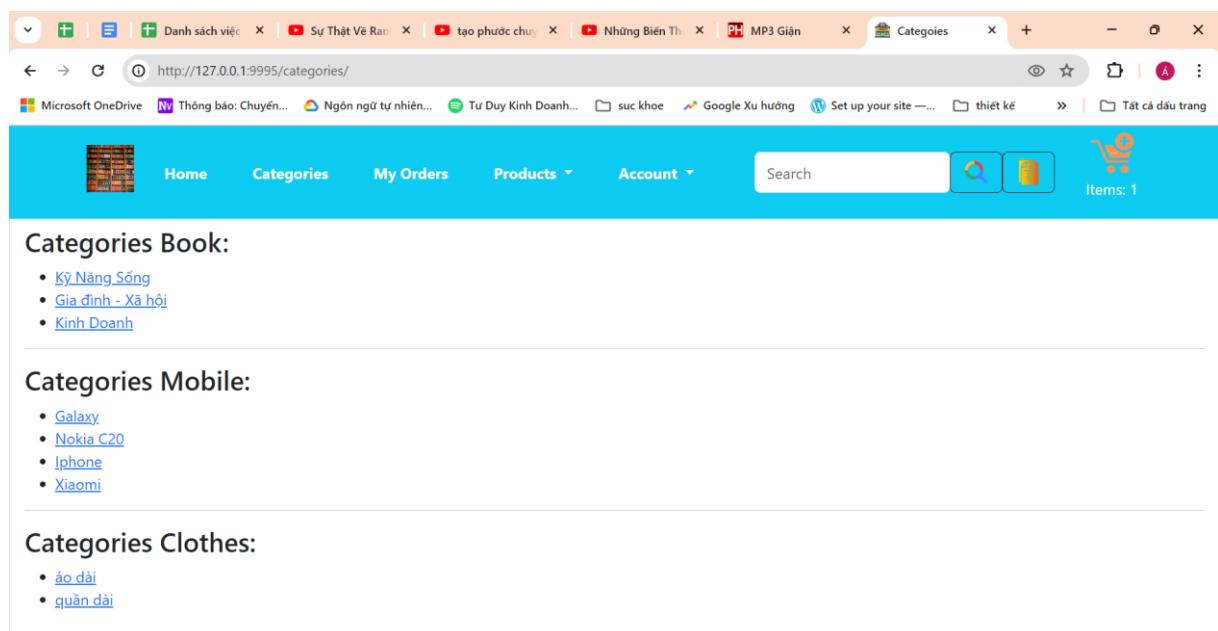


Change Password

Password:

Confirm Password:

Categories



Categories Book:

- [Kỹ Năng Sống](#)
- [Gia đình - Xã hội](#)
- [Kinh Doanh](#)

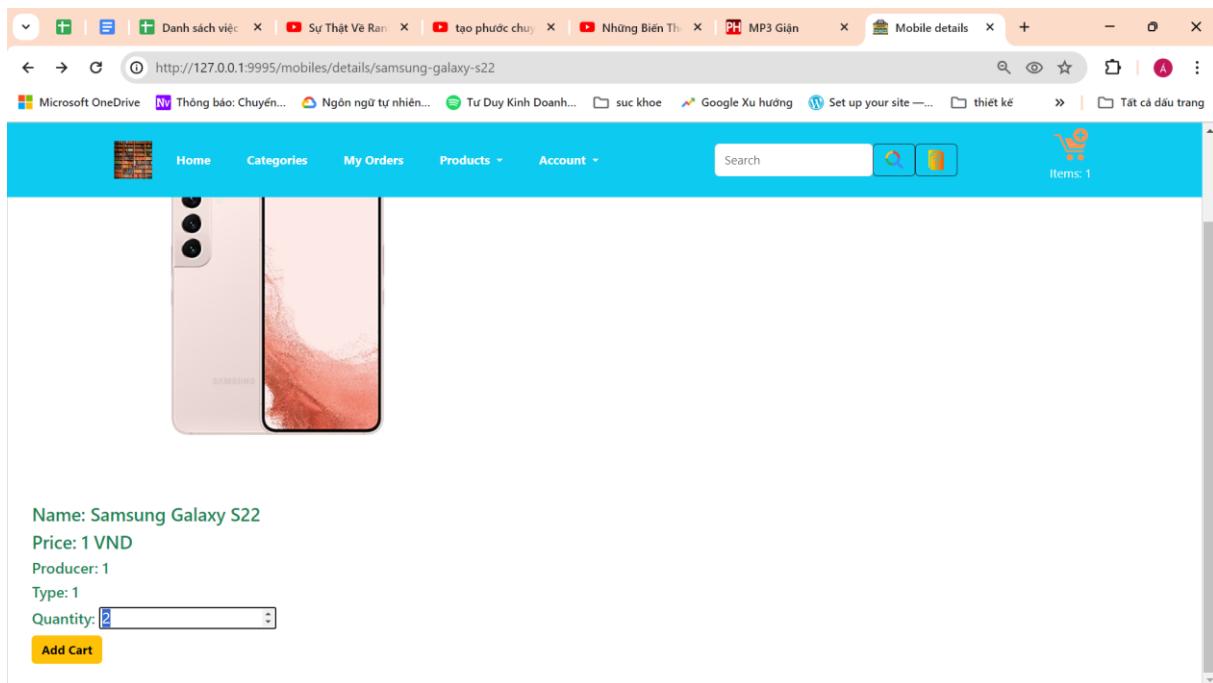
Categories Mobile:

- [Galaxy](#)
- [Nokia C20](#)
- [Iphone](#)
- [Xiaomi](#)

Categories Clothes:

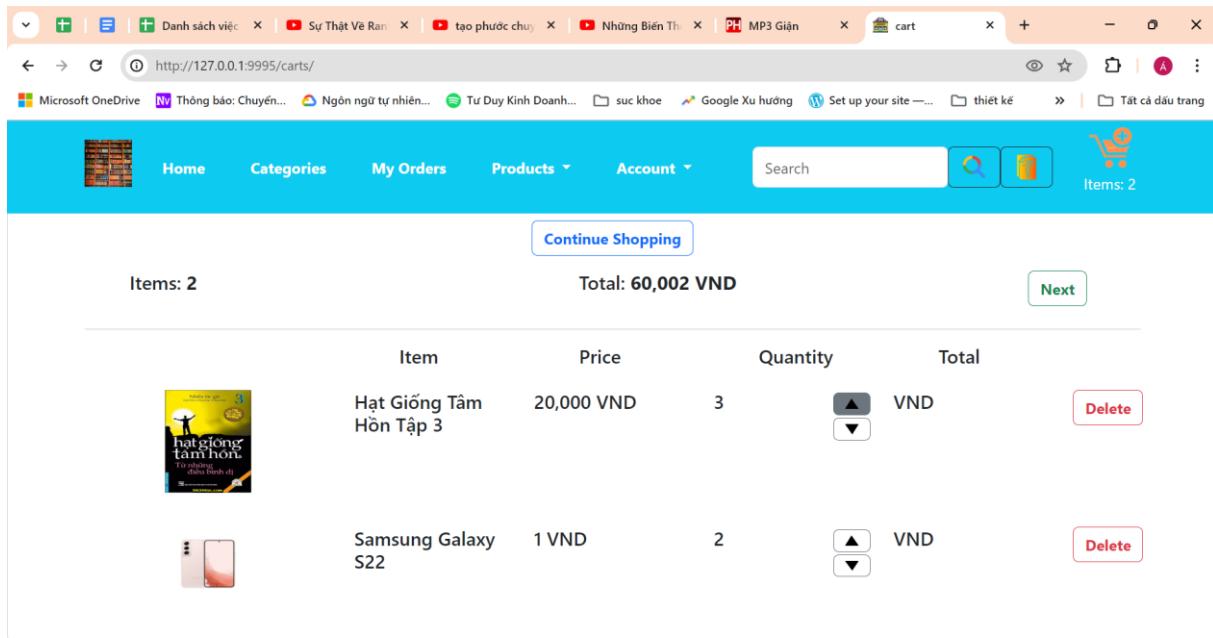
- [áo dài](#)
- [quần dài](#)

View detail product và add to cart



Name: Samsung Galaxy S22
Price: 1 VND
Producer: 1
Type: 1
Quantity:
Add Cart

View cart



Items: 2 Total: 60,002 VND **Next**

| Item | Price | Quantity | Total |
|---|------------|----------|---|
|  Hạt Giống Tâm Hồn Tập 3 | 20,000 VND | 3 |  VND Delete |
|  Samsung Galaxy S22 | 1 VND | 2 |  VND Delete |

Update items in cart

Items: 1

Total: 80,000 VND

| Item | Price | Quantity | Total |
|-------------------------|------------|----------|-------|
| Hạt Giống Tâm Hồn Tập 3 | 20,000 VND | 4 | VND |

Delete

Nhập thông tin shipment

Name: anh le

Phone: 0123456789

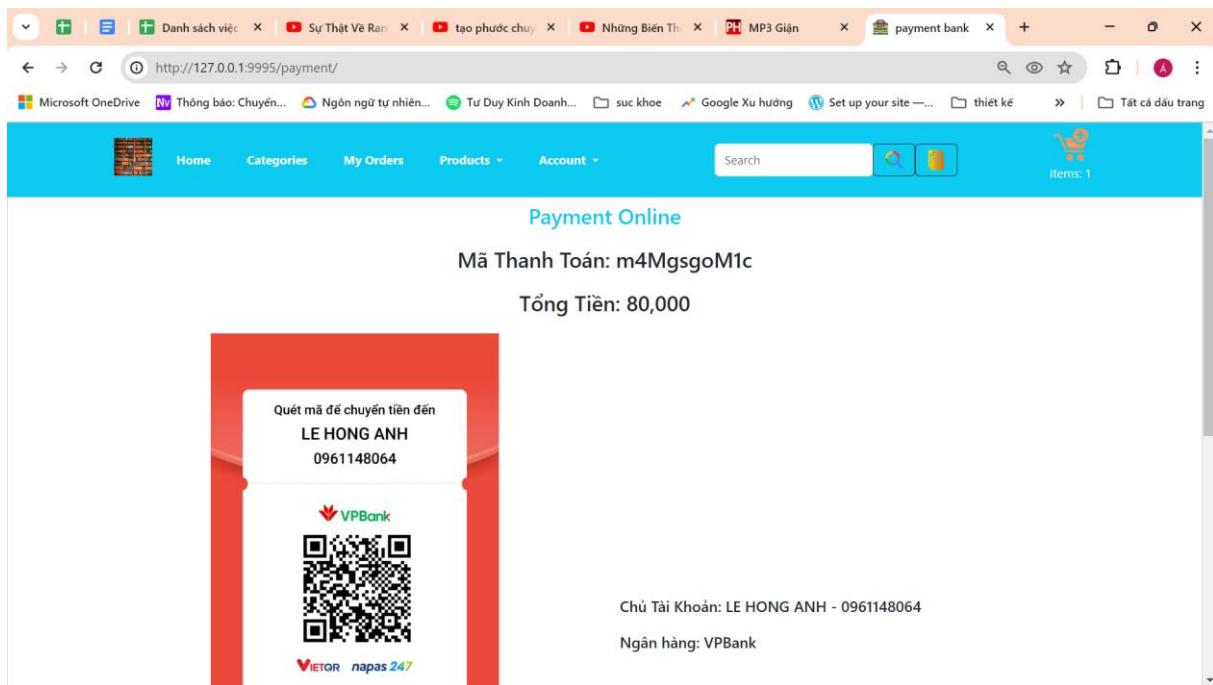
Email: lea81807@gmail.com

Address: Ha Noi

City: Ha Noi

Select Method Payment

Thực hiện thanh toán online



Payment Online

Mã Thanh Toán: m4MgsgoM1c

Tổng Tiền: 80,000

Quét mã để chuyển tiền đến
LE HONG ANH
0961148064

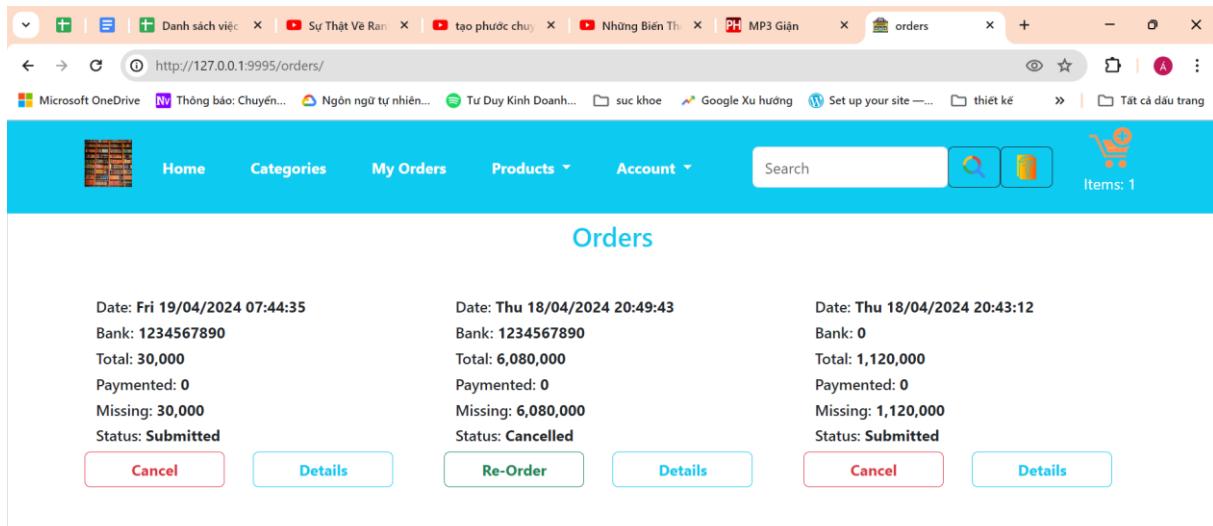
VPBank

Chủ Tài Khoản: LE HONG ANH - 0961148064

Ngân hàng: VPBank

VIETQR napas 247

View list ordered



Orders

| Date | Bank | Total | Paymented | Missing | Status | Actions |
|-------------------------|------------|-----------|-----------|-----------|-----------|--|
| Fri 19/04/2024 07:44:35 | 1234567890 | 30,000 | 0 | 30,000 | Submitted | Cancel Details |
| Thu 18/04/2024 20:49:43 | 1234567890 | 6,080,000 | 0 | 6,080,000 | Cancelled | Re-Order Details |
| Thu 18/04/2024 20:43:12 | 0 | 1,120,000 | 0 | 1,120,000 | Submitted | Cancel Details |

View details an order

http://127.0.0.1:9995/orders/details/21

Microsoft OneDrive Thông báo: Chuyển... Ngôn ngữ tự nhiên... Tú Duy Kinh Doanh... suc khoe Google Xu hướng Set up your site —... thiết kế Tất cả dấu trang

Home Categories My Orders Products Account Search Items: 1

Name: anh
Phone: 1234567890
Address: ha dong
Email: lea81807@gmail.com

Products Ordered

Items: 1 Total: 30,000 VND

| Item | Price | Quantity | Total |
|------|------------|----------|------------|
| | 30,000 VND | 1 | 30,000 VND |

Tìm kiếm product

http://127.0.0.1:9995/search/result/?keyword=tôi+học

Microsoft OneDrive Thông báo: Chuyển... Ngôn ngữ tự nhiên... Tú Duy Kinh Doanh... suc khoe Google Xu hướng Set up your site —... thiết kế Tất cả dấu trang

Home Categories My Orders Products Account Search Items: 1

result search for product "tôi học"

Books

Tôi Tự Học
Add Cart View
20000 VND

Login với account admin

http://127.0.0.1:9995

Microsoft OneDrive Thông báo: Chuyển... Ngôn ngữ tự nhiên... Tự Duy Kinh Doanh... suc khoe Google Xu hướng Set up your site —... thiết kế Tất cả dấu trang

Home Reports Manage Orders Account

Hello Admin: anhgdt

Xem báo cáo những sản phẩm được mua nhiều

http://127.0.0.1:9995/manager/report-product

Microsoft OneDrive Thông báo: Chuyển... Ngôn ngữ tự nhiên... Tự Duy Kinh Doanh... suc khoe Google Xu hướng Set up your site —... thiết kế Tất cả dấu trang

| Item | Price | Quantity | Total |
|---|---------------|----------|----------------|
|  iPhone 15 128GB | 3,000,000 VND | 13 | 39,000,000 VND |
|  Nokia C20 | 1,000,000 VND | 3 | 3,000,000 VND |
|  CÔ HỌC TINH HOA | 20,000 VND | 5 | 100,000 VND |
|  Bộ quần áo adidas Nam GS8907 | 100,000 VND | 1 | 100,000 VND |
|  Mè - Biểu Hiện Của Tình Thương | 30,000 VND | 1 | 30,000 VND |

Xem báo cáo khách hàng theo chi tiêu

| Username | Email | Phone | Created at | Total |
|----------|--------------------|------------|-------------------------|----------------|
| customer | customer@gmail.com | 1111222233 | Fri 19/04/2024 19:15:13 | 35,000,000 VND |
| leanh | leanh@gmail.com | 0123456789 | Fri 12/04/2024 08:34:50 | 7,230,000 VND |

